Fast and Succinct Population Protocols for Presburger Arithmetic

Philipp Czerner
Department of Informatics, TU Munich, Germany

Roland Guttenberg
Department of Informatics, TU Munich, Germany

Martin Helfrich
Department of Informatics, TU Munich, Germany

Javier Esparza
Department of Informatics, TU Munich, Germany

Abstract

In their 2006 seminal paper in Distributed Computing, Angluin et al. present a construction that, given any Presburger predicate as input, outputs a leaderless population protocol that decides the predicate. The protocol for a predicate of size m (when expressed as a Boolean combination of threshold and remainder predicates with coefficients in binary) runs in $\mathcal{O}(m \cdot n^2 \log n)$ expected number of interactions, which is almost optimal in n, the number of interacting agents. However, the number of states of the protocol is exponential in m. This is a problem for natural computing applications, where a state corresponds to a chemical species and it is difficult to implement protocols with many states. Blondin et al. described in STACS 2020 another construction that produces protocols with a polynomial number of states, but exponential expected number of interactions. We present a construction that produces protocols with $\mathcal{O}(m)$ states that run in expected $\mathcal{O}(m^7 \cdot n^2)$ interactions, optimal in n, for all inputs of size $\Omega(m)$. For this, we introduce population computers, a carefully crafted generalization of population protocols easier to program, and show that our computers for Presburger predicates can be translated into fast and succinct population protocols.

2012 ACM Subject Classification Theory of computation → Distributed computing models

Keywords and phrases population protocols, fast, succinct, population computers

Funding This work was supported by an ERC Advanced Grant (787367: PaVeS) and by the Research Training Network of the Deutsche Forschungsgemeinschaft (DFG) (378803395: ConVeY).

1 Introduction

Population protocols [4, 5] are a model of computation in which indistinguishable, mobile finite-state agents, randomly interact in pairs to decide whether their initial configuration satisfies a given property, modelled as a predicate on the set of all configurations. The decision is taken by *stable consensus*; eventually all agents agree on whether the property holds or not, and never change their mind again. Population protocols are very close to chemical reaction networks, a model in which agents are molecules and interactions are chemical reactions.

In a seminal paper, Angluin *et al.* proved that population protocols decide exactly the predicates definable in Presburger arithmetic (PA) [7]. One direction of the result is proved in [5] by means of a construction that takes as input a Presburger predicate and outputs a protocol that decides it. The construction uses the quantifier elimination procedure for PA: every Presburger formula φ can be transformed into an equivalent boolean combination of *threshold* predicates of the form $\vec{a} \cdot \vec{x} \geq c$ and *remainder* predicates of the form $\vec{a} \cdot \vec{x} \equiv_m c$, where $\vec{a}$ is an integer vector, c and m are integers, and $\equiv_m$ denotes congruence modulo m [13]. Slightly abusing language, we call the set of these boolean combinations *quantifier-free*

Presburger arithmetic (QFPA).[1] Using that PA and QFPA have the same expressive power, Angluin *et al.* first construct protocols for all threshold and remainder predicates, and then show that the predicates computed by protocols are closed under negation and conjunction.

The two fundamental parameters of a protocol are the expected number of interactions until a stable consensus is reached, and the number of states of each agent. The expected number of interactions divided by the number of agents, also called the parallel execution time, is an adequate measure of the runtime of a protocol when interactions occur in parallel according to a Poisson process [6]. The number of states measures the complexity of an agent. In natural computing applications, where a state corresponds to a chemical species, it is difficult to implement protocols with many states.

Given a formula φ of QFPA, let m be the number of bits of the largest coefficient of φ in absolute value, and let s be the number of atomic formulas of φ, respectively. Let n be the number of agents participating in the protocol. The construction of [5] yields a protocol with $\mathcal{O}(s \cdot n^2 \log n)$ expected interactions. Observe that the protocol does not have a leader (an auxiliary agent helping the other agents to coordinate), and agents have a fixed number of states, independent of the size of the population. Under these assumptions, which are also the assumptions of this paper, every protocol for the majority predicate needs $\Omega(n^2)$ expected interactions [1], and so the construction is nearly optimal.[2] However, the number of states is $\Omega(2^{m+s})$, or $\Omega(2^{|\varphi|})$ in terms of the number $|\varphi|$ of bits needed to write φ with coefficients in binary. This is well beyond the only known lower bound, showing that for every construction there exist an infinite subset of predicates φ for which the construction produces protocols with $\Omega(|\varphi|^{1/4})$ states [9]. So the constructions of [5], and also those of [6, 3, 12], produce *fast* but *very large* protocols.

In [9, 8] Blondin *et al.* exhibit a construction that produces *succinct* protocols with $\mathcal{O}(\text{poly}(|\varphi|))$ states. However, they do not analyse their stabilisation time. We demonstrate that they run in $\Omega(2^n)$ expected interactions. Loosely speaking, the reason is the use of transitions that “revert” the effect of other transitions. This allows the protocol to “try out” different distributions of agents, retracing its steps until it hits the right one, but also makes it very slow. So [9, 8] produce *succinct* but *very slow* protocols.

Is it possible to produce protocols that are both *fast* and *succinct*? We give an affirmative answer. We present a construction that yields for every formula φ of QFPA a protocol with $\mathcal{O}(\text{poly}(|\varphi|))$ states and $\mathcal{O}(\text{poly}(|\varphi|) \cdot n^2)$ expected interactions. So our construction achieves optimal stabilisation time in n, and, at the same time, yields more succinct protocols than the construction of [8]. Moreover, for inputs of size $\Omega(|\varphi|)$ (a very mild constraint when agents are molecules), we obtain protocols with $\mathcal{O}(|\varphi|)$ states.

Our construction relies on *population computers*, a carefully crafted generalization of the population protocol model of [5]. Population computers extend population protocols in three ways. First, they can exhibit certain *k-way interactions* between more than two agents. Second, they have a more flexible *output condition*, defined by an arbitrary function that assigns an output to every subset of states, instead of to every state.[3] Finally, population computers can use *helpers*: auxiliary agents that, like leaders, help regular agents to coordinate themselves but whose number, contrary to leaders, is not known *a priori*. We exhibit succinct population computers for all Presburger predicates in which every run is finite, and show

---

[1] Remainder predicates cannot be directly expressed in Presburger arithmetic without quantifiers.
[2] If the model is extended by allowing a *leader* (and one considers the slightly weaker notion of convergence time), or the number of states of an agent is allowed to grow with the population size, $\mathcal{O}(n \cdot \text{polylog}(n))$ interactions can be achieved [6, 3, 2, 12, 11].
[3] Other output conventions for population protocols have been considered [10].

how to translate such population computers into fast and succinct population protocols.

Organization of the paper. We give preliminary definitions in Section 2 and introduce population computers in Section 3. Section 4 gives an overview of the rest of the paper and summarises our main results. Section 5 describes why previous constructions were either not succinct or slow. Section 6 describes population computers for every Presburger predicate. Section 7 converts these computers into succinct population protocols. Section 8 shows that the resulting protocols are also fast.

Appendix A completes the proofs of Section 5. For the other appendices, there is no one-to-one correspondence to sections of the main paper, instead they are grouped by the construction they analyse. Appendix B concerns the construction of Section 6, but also analyses speed. The four parts of our conversion process are analysed separately, in Appendices C, D, E and F. Appendix G combines the previous to prove the complete conversion theorem. Appendix H summarises the definitions for our speed analyses, and Appendix I contains minor technical lemmata.

2 Preliminaries

Multisets. Let E be a finite set. A multiset over E is a mapping $E \to \mathbb{N}$, and $\mathbb{N}^E$ denotes the set of all multisets over E. We sometimes write multisets using set-like notation, e.g. $\wr a, 2 \cdot b \wr$ denotes the multiset v such that $v(a) = 1$, $v(b) = 2$ and $v(e) = 0$ for every $e \in E \setminus \{a, b\}$. The empty multiset $\wr\wr$ is also denoted $\emptyset$.

For $E' \subseteq E$, $v(E') := \sum_{e \in E'} v(e)$ is the number of elements in v that are in E'. The *size* of $v \in \mathbb{N}^E$ is $|v| := v(E)$. The *support* of $v \in \mathbb{N}^E$ is the set $\text{supp}(v) := \{e \in E \mid v(e) > 0\}$. If $E \subseteq \mathbb{Z}$, then we let $\text{sum}(v) := \sum_{e \in E} e \cdot v(e)$ denote the sum of all the elements of v. Given $u, v \in \mathbb{N}^E$, $u + v$ and $u - v$ denote the multisets given by $(u + v)(e) := u(e) + v(e)$ and $(u - v)(e) := u(e) - v(e)$ for every $e \in E$. The latter is only defined if $u \geq v$.

Multiset rewriting transitions. A *multiset rewriting transition*, or just a *transition*, is a pair $(r, s) \in \mathbb{N}^E \times \mathbb{N}^E$, also written $r \mapsto s$. A transition $t = (r, s)$ is *enabled* at $v \in \mathbb{N}^E$ if $v \geq r$, and its *occurrence* leads to $v' := v - r + s$, denoted $v \to_t v'$. We call $v \to_t v'$ a *step*. The multiset v is *terminal* if it does not enable any transition. An *execution* is a finite or infinite sequence $v_0, v_1, ...$ of multisets such that $v \to_{t_1} v_1 \to_{t_2} \cdots$ for some sequence $t_1, t_2, ...$ of transitions. A multiset v' is *reachable* from v if there is an execution $v_0, v_1, ..., v_k$ with $v_0 = v$ and $v_k = v'$; we also say that the execution *leads from v to v'*. An execution is a *run* if it is infinite or it is finite and its last multiset is terminal. A run $v_0, v_1, ...$ is *fair* if it is finite, or it is infinite and for every multiset v, if v is reachable from v_i for infinitely many $i \geq 0$, then $v = v_j$ for some $j \geq 0$.

Presburger arithmetic Angluin *et al.* proved that population protocols decide exactly the predicates $\mathbb{N}^k \to \{0, 1\}$ definable in Presburger arithmetic, the first-order theory of addition, which coincide with the *semilinear* predicates [13]. Using the quantifier elimination procedure of Presburger arithmetic, every Presburger predicate can be represented as a Boolean combination of *threshold* and *remainder* predicates. A predicate $\varphi : \mathbb{N}^v \to \{0, 1\}$ is a threshold predicate if $\varphi(x_1, ..., x_v) = (\sum_{i=1}^{v} a_i x_i \geq c)$, where $a_1, ..., a_v, c \in \mathbb{Z}$, and a remainder predicate if $\varphi(x_1, ..., x_v) = (\sum_{i=1}^{v} a_i x_i \equiv_m c)$, where $a_1, ..., a_v \in \mathbb{Z}$, $m \geq 1$, $c \in \{0, ..., m-1\}$, and $a \equiv_m b$ denotes that a is congruent to b modulo m. We call the set of these formulas *quantifier-free Presburger arithmetic*, or QFPA. The *size* of a predicate is the minimal number of bits of a formula of QFPA representing it, with coefficients written in binary.

3 Population Computers

Population computers are a generalization of population protocols that allows us to give very concise descriptions of our protocols for Presburger predicates.

Syntax. A *population computer* is a tuple $\mathcal{P} = (Q, \delta, I, O, H)$, where:

- Q is a finite set of *states*. Multisets over Q are called *configurations*.
- $\delta \subseteq \mathbb{N}^Q \times \mathbb{N}^Q$ is a set of multiset rewriting transitions $r \mapsto s$ over Q such that $|r| = |s| \geq 2$ and $|\text{supp}(r)| \leq 2$. Further, we require that δ is a partial function, so $s_1 = s_2$ for all r, s_1, s_2 with $(r_1 \mapsto s_1), (r_2 \mapsto s_2) \in \delta$. A transition $r \mapsto s$ is *binary* if $|r| = 2$. We call a population computer is *binary* if every transition binary.
- $I \subseteq Q$ is a set of *input states*. An *input* is a configuration C such that $\text{supp}(C) \subseteq \text{supp}(I)$.
- $O : 2^Q \to \{0, 1, \bot\}$ is an *output function*. The *output* of a configuration C is $O(\text{supp}(C))$. An output function O is a *consensus output* if there is a partition $Q = Q_0 \cup Q_1$ of Q such that $O(Q') = 0$ iff $Q' \subseteq Q_0$, $O(Q') = 1$ iff $Q' \subseteq Q_1$, and $O(Q') = \bot$ otherwise.
- $H \in \mathbb{N}^{Q \setminus I}$ is a multiset of *helper agents* or just *helpers*. A *helper configuration* is a configuration C such that $\text{supp}(C) \subseteq \text{supp}(H)$ and $C \geq H$.

Graphical notation. We visualise population computers as Petri nets (see e.g. Figure 3). Places (circles) and transitions (squares) represent respectively states and transitions. To visualise configurations, we draw agents as tokens (smaller filled circles).

Semantics. Intuitively, a population computer decides which output (0 or 1) corresponds to an input C_I as follows. It adds to the agents of C_I an arbitrary helper configuration C_H of agents to produce the initial configuration $C_I + C_H$. Then it starts the computation and lets it stabilise to configurations of output 1 or output 0. Formally, the *initial configurations* of $\mathcal{P}$ for input C_I are all configurations of the form $C_I + C_H$ for some helper configuration C_H. A run $C_0C_1...$ *stabilises to* b if there exists an $i \geq 0$ such that $O(\text{supp}(C_i)) = b$ and C_i only reaches configurations C' with $O(\text{supp}(C')) = b$. An input C_I *has output* b if for every initial configuration $C_0 = C_I + C_H$, every fair run starting at C_0 stabilises to b. A population computer $\mathcal{P}$ *decides* a predicate $\varphi : \mathbb{N}^I \to \{0, 1\}$ if every input C_I has output $\varphi(C_I)$.

Terminating and bounded computers. A population computer is *bounded* if no run starting at an initial configuration C is infinite, and *terminating* if no fair run starting at C is infinite. Observe that bounded population computers are terminating.

Size and adjusted size. Let $\mathcal{P} = (Q, \delta, I, O, H)$ be a population computer. We assume that O is described as a boolean circuit with size(O) gates. For every transition $t = (r \mapsto s)$ let $|t| := |r|$. The *size* of $\mathcal{P}$ is $\text{size}(\mathcal{P}) := |Q| + |H| + \text{size}(O) + \sum_{t \in \delta} |t|$. If $\mathcal{P}$ is binary, then (as for population protocols) we do not count the transitions and define the *adjusted size* $\text{size}_2(\mathcal{P}) := |Q| + |H| + \text{size}(O)$. Observe that both the size of a transition and the size of the helper multiset are the number of elements, i.e. the size in unary, strengthening our later result about the existence of succinct population computers.

Population protocols. A population computer $\mathcal{P} = (Q, \delta, I, O, H)$ is a *population protocol* if it is binary, has no helpers ($H = \emptyset$), and O is a consensus output. It is easy to see that this definition coincides with the one of [5]. The speed of a binary population computer with no helpers, and so in particular of a population protocol, is defined as follows. We assume a probabilistic execution model in which at configuration C two agents are picked uniformly at random and execute a transition, if possible, moving to a configuration C' (by assumption they enable at most one transition). This is called an *interaction*. Repeating this process, we generate a *random execution* $C_0C_1...$. We say that the execution *stabilises* at time t if C_t

reaches only configurations C' with $O(\text{supp}(C')) = O(\text{supp}(C_t))$, and we say that $\mathcal{P}$ *decides* φ *within* T *interactions* if it decides φ and $\mathbb{E}(t) \leq T$. See e.g. [6] for more details.

Population computers vs. population protocols. Population computers generalise population protocols in three ways:

- They have non-binary transitions, but only those in which the interacting agents populate at most two states. For example, $⦃p, p, q⦄ \mapsto ⦃p, q, o⦄$ (which in the following is written simply as $p, p, q \mapsto p, q, o$) is allowed, but $p, q, o \mapsto p, p, q$ is not.
- They use a multiset H of auxiliary helper agents, but the addition of more helpers does not change the output of the computation. Intuitively, contrary to the case of leaders, agents do not know any upper bound on the number of helpers, and so the protocol cannot rely on this bound for correctness or speed.
- They have a more flexible output condition. Loosely speaking, population computers accept by stabilising the population to an accepting set of states, instead of to a set of accepting states.

4 Overview and Main Results

Given a predicate $\varphi \in QFPA$ over variables $x_1, ..., x_v$, the rest of this paper shows how to construct a fast and succinct population protocol deciding φ. First, Section 5 gives an overview of previous constructions and explains why they are not fast or not succinct. Then we proceed in five steps:

1. Construct the predicate $\text{double}(\varphi) \in QFPA$ over variables $x_1, ..., x_v, x'_1, ..., x'_v$ by syntactically replacing every occurrence of x_i in φ by $x_i + 2x'_i$. For example, if $\varphi = (x - y \geq 0)$ then $\text{double}(\varphi) = (x + 2x' - y - 2y' \geq 0)$. Observe that $|\text{double}(\varphi)| \in \mathcal{O}(|\varphi|)$.
2. Construct a succinct bounded population computer $\mathcal{P}$ deciding $\text{double}(\varphi)$.
3. Convert $\mathcal{P}$ into a succinct population protocol $\mathcal{P}'$ deciding φ for inputs of size $\Omega(|\varphi|)$.
4. Prove that $\mathcal{P}'$ runs within $\mathcal{O}(n^3)$ interactions.
5. Use a refined running-time analysis to prove that $\mathcal{P}'$ runs within $\mathcal{O}(n^2)$ interactions.

Section 6 constructs bounded population computers for all predicates $\varphi \in QFPA$. This allows us to conduct steps 1 and 2. More precisely, the section proves:

▶ **Theorem 1.** *For every predicate $\varphi \in QFPA$ there exists a bounded population computer of size $\mathcal{O}(|\varphi|)$ that decides φ.*

Section 7 proves the following conversion theorem (steps 3 and 4).

▶ **Theorem 2.** *Every bounded population computer of size m deciding* $\text{double}(\varphi)$ *can be converted into a terminating population protocol with $\mathcal{O}(m^2)$ states which decides φ in at most $\mathcal{O}(f(m)\, n^3)$ interactions for inputs of size $\Omega(m)$, for some function f.*

Section 8 introduces α*-rapid* population computers, where $\alpha \geq 1$ is a certain parameter, and uses a more detailed analysis to show that the population protocols of Theorem 2 are in fact smaller and faster (step 5):

▶ **Theorem 3.** **(a)** *The population computers constructed in Theorem 1 are $\mathcal{O}(|\varphi|^3)$-rapid.*

(b) *Every α-rapid population computer of size m deciding* $\text{double}(\varphi)$ *can be converted into a terminating population protocol with $\mathcal{O}(m)$ states that decides φ in $\mathcal{O}(\alpha\, m^4 n^2)$ interactions for inputs of size $\Omega(m)$.*

The restriction to inputs of size $\Omega(m)$ is very mild. Moreover, it can be lifted using a technique of [8], at the price of adding additional states (and at no cost regarding asymptotic speed, since the speed of the new protocol only changes for inputs of size $\mathcal{O}(m)$):

▶ **Corollary 4.** *For every $\varphi \in$ QFPA there exists a terminating population protocol with $\mathcal{O}(\text{poly}(|\varphi|))$ states that decides φ in $\mathcal{O}(f(|\varphi|)\, n^2)$ interactions, for a function f.*

It is known that the majority predicate can only be decided in $\Omega(n^2)$ interactions by population protocols [1], so — as a general construction — our result is optimal w.r.t. time. Regarding space, an $\Omega(|\varphi|^{1/4})$ lower bound was shown in [9], leaving a polynomial gap.

5 Previous Constructions: Angluin *et al.* and Blondin *et al.*

The population protocols for a quantifier free Presburger predicate φ constructed in [5] are not *succinct*, i.e. do not have $\mathcal{O}(|\varphi|^a)$ states for any constant a, and those of [8] are not *fast*, i.e. do not have speed $\mathcal{O}(|\varphi|^a n^b)$ for any constants a, b. We explain why with the help of some examples.

▶ **Example 5.** Consider the protocol of [5] for the predicate $\varphi = (x - y \geq 2^d)$. The states are the triples (ℓ, b, u) where $\ell \in \{A, P\}$, $b \in \{Y, N\}$ and $-2^d \leq u \leq 2^d$. Intuitively, ℓ indicates whether the agent is active (A) or passive (P), b indicates whether it currently believes that φ holds (Y) or not (N), and u is the agent's wealth, which can be negative. Agents for input x are initially in state $(A, N, 1)$, and agents for y in $(A, N, -1)$. If two passive agents meet their encounter has no effect. If at least one agent is active, then the result of the encounter is given by the transition $(*, *, u), (*, *, u') \mapsto (A, b, q), (P, b, r)$ where $b = Y$ if $u + u' \geq 2^d$ else N; $q = \max(-2^d, \min(2^d, u + u'))$; and $r = (u + u') - q$. The protocol stabilises after $\mathcal{O}(n^2 \log n)$ expected interactions [5], but it has $2^{d+1} + 1$ states, exponentially many in $|\varphi| \in \Theta(d)$.

▶ **Example 6.** We give a protocol for $\varphi = (x - y \geq 2^d)$ with a polynomial number of states. This is essentially the protocol of [8]. We remove states and transitions from the protocol of Example 5, retaining only the states (ℓ, b, u) such that u is a power of 2, and some of the transitions involving these states:

$$\begin{array}{rcll}
(*, *, 2^i), (*, *, 2^i) & \mapsto & (A, N, 2^{i+1}), (P, N, 0) & \text{for every } 0 \leq i \leq d-2 \\
(*, *, 2^{d-1}), (*, *, 2^{d-1}) & \mapsto & (A, Y, 2^d), (P, Y, 0) & \\
(*, *, -2^i), (*, *, -2^i) & \mapsto & (A, N, -2^{i+1}), (P, N, 0) & \text{for every } 0 \leq i \leq d-1 \\
(*, *, 2^i), (*, *, -2^i) & \mapsto & (A, N, 0), (P, N, 0) & \text{for every } 0 \leq i \leq d-1
\end{array}$$

The protocol is not yet correct. For example, for $d = 1$ and the input $x = 2, y = 1$, the protocol can reach in one step the configuration in which the three agents (two x-agents and one y-agent) are in states $(A, Y, 2), (P, Y, 0), (A, N, -1)$, after which it gets stuck. In [8] this is solved by adding "reverse" transitions:

$$\begin{array}{rcll}
(A, N, 2^{i+1}), (P, N, 0) & \mapsto & (A, N, 2^i), (P, N, 2^i) & \text{for every } 0 \leq i \leq d-2 \\
(A, Y, 2^d), (P, Y, 0) & \mapsto & (A, N, 2^{d-1}), (P, N, 2^{d-1}) & \\
(A, N, -2^{i+1}), (P, N, 0) & \mapsto & (A, N, -2^i), (A, N, -2^i) & \text{for every } 0 \leq i \leq d-1
\end{array}$$

The protocol has only $\Theta(d)$ states and transitions, but runs within $\Omega(n^{2^d-2})$ interactions. Consider the inputs x, y such that $x - y = 2^d$, and let $n := x + y$. Say that an agent is *positive* at a configuration if it has positive wealth at it. The protocol can only stabilise if it reaches a configuration with exactly one positive agent with wealth 2^d. Consider a configuration with $i < 2^d$ positive agents. The next configuration can have $i-1$, i, or $i+1$ positive agents. The probability of $i+1$ positive agents is $\Omega(1/n)$, while that of $i-1$ positive agents is only $\mathcal{O}(1/n^2)$, and the expected number of interactions needed to go from 2^d positive agents to only 1 is $\Omega(n^{2^d-1})$ Appendix A.1.

▶ **Example 7.** Given protocols $\mathcal{P}_1, \mathcal{P}_2$ with n_1 and n_2 states deciding predicates φ_1 and φ_2, Angluin et al. construct in [5] a protocol $\mathcal{P}$ for $\varphi_1 \wedge \varphi_2$ with $n_1 \cdot n_2$ states. It follows that the number of states of a protocol for $\varphi := \varphi_1 \wedge \cdots \wedge \varphi_s$ grows exponentially in s, and so in $|\varphi|$. Blondin *et al.* give an alternative construction with polynomially many states [8, Section 5.3]. However, their construction contains transitions that, as in the previous example, reverse the effect of other transitions, and make the protocol very slow. The problem is already observed in the toy protocol with states q_1, q_2 and transitions $q_1, q_1 \mapsto q_2, q_2$ and $q_1, q_2 \mapsto q_1, q_1$. (Similar transitions are used in the initialisation of [8].) Starting with an even number $n \geq 2$ of agents in q_1, eventually all agents move to q_2 and stay there, but the expected number of interactions is $\Omega(2^{n/10})$ Appendix A.2.

6 Succinct Bounded Population Computers for Presburger Predicates

In Sections 6.1 and 6.2 we construct population computers for remainder and threshold predicates in which all coefficients are powers of two. We present the remainder case in detail, and sketch the threshold case. The generalization to arbitrary coefficients is achieved by means of a gadget very similar to the one we used to compute boolean combinations of predicates. This later gadget is presented in Section 6.3, and so we introduce the generalization there.

6.1 Population computers for remainder predicates

Let $Pow^+ = \{2^i \mid i \geq 0\}$ be the set of positive powers of 2.

We construct population computers $\mathcal{P}_\varphi$ for remainder predicates $\varphi := \sum_{i=1}^{v} a_i x_i \equiv_m c$, where $a_i \in Pow^+ \cap \{0, ..., m-1\}$ for every $1 \leq i \leq v$, $m \in \mathbb{N}$, and $c \in \{0, ..., m-1\}$. We say that a finite multiset r over Pow^+ *represents* the residue $\text{rep}(r) := \text{sum}(r) \bmod m$. For example, if $m = 11$ then $r_{18} := ⦃2^3, 2^3, 2^1⦄$ represents 7. Accordingly, we call the multisets over Pow^+ *representations*. A *representation of degree d* only contains elements of $Pow_d^+ := \{2^d, 2^{d-1}, ..., 2^0\}$. A representation r is a *support representation* if $r(x) \leq 1$ for every $x \in Pow^+$; so its represented value is completely determined by the support. For example, r_{18} is not a support representation of 7, but $⦃2^5, 2^3⦄$ is.

We proceed to construct $\mathcal{P}_\varphi$. Let us give some intuition first. $\mathcal{P}_\varphi$ has $Pow_d^+ \cup \{0\}$ as set of states. We extend the notion of representation to configurations by disregarding agents in state 0; a configuration is therefore a support representation if all states except 0 have at most one agent. The initial states of $\mathcal{P}_\varphi$ are chosen so that every initial configuration for an input $(x_1, ..., x_v)$ is a representation of the residue $z := \sum_{i=1}^{v} a_i x_i \bmod m$. The transitions transform this initial representation of z into a support representation of z. Whether $z \equiv_m c$ holds or not depends only on the support of this representation, and the output function thus returns 1 for the supports satisfying $z \equiv_m c$, and 0 otherwise. Let us now formally describe $\mathcal{P}_\varphi$ for $\varphi := \sum_{i=1}^{v} a_i x_i \equiv_m c$ where $a_i \in Pow^+ \cap \{0, ..., m-1\}$.

States and initial states. Let $d := \lceil \log_2 m \rceil$. The set of states is $Q = Pow_d^+ \cup \{0\}$. The set of initial states is $I := \{a_1, ..., a_v\}$. Observe that an input $C_I = ⦃x_1 \cdot a_1, ..., x_v \cdot a_v⦄$ is a representation of z, but not necessarily a support representation.

Transitions. Transitions ensure that non-support representations, i.e. representations with two or more agents in some state q, are transformed into representations of the same residue

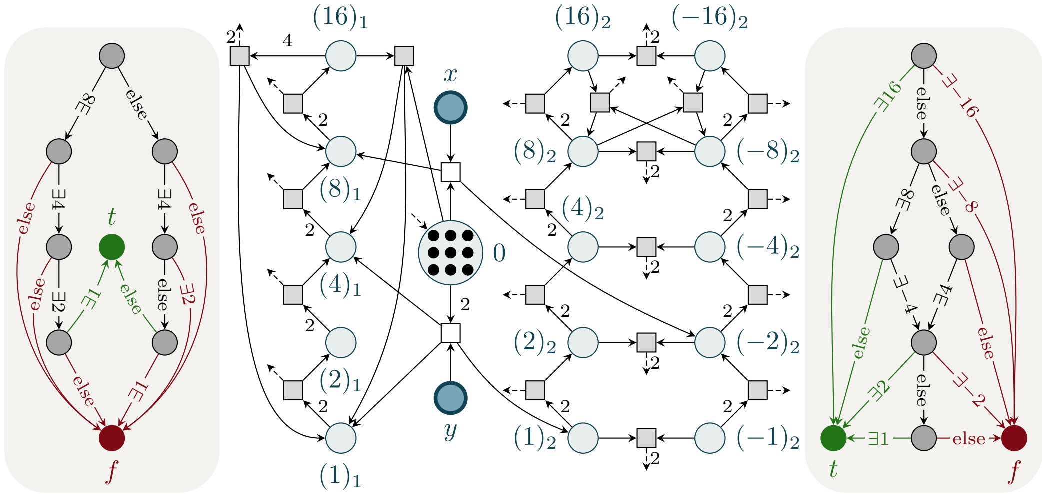

Figure 1 (middle) Graphical Petri net representation (see Section 3) of population computer for the predicate $\varphi_1 \vee \varphi_2$ with $\varphi_1 = (8x + 5y \equiv_{11} 4)$ and $\varphi_2 = (y - 2x \geq 5)$. All dashed arrows implicitly lead to the reservoir state 0. It has 22 helpers although only 9 are drawn for space reasons. (left) decision diagram for output function of remainder predicate $8x + 5y \equiv_{11} 4$. It checks if the total value is 15 or 4. Starting at the top node of the diagram: if state 8 is populated, we move to the left child, otherwise to the right child; at the left child, if state 4 is populated we move to the right child, etc. (right) decision diagram for output function of threshold predicate $y - 2x \geq 5$.

"closer" to a support representation. For $q \in 2^0, ..., 2^{d-1}$ we introduce the transition:

$$2^i, 2^i \quad \mapsto \quad 2^{i+1}, 0 \qquad \text{for } 0 \leq i \leq d-1 \qquad \langle\text{combine}\rangle$$

For $q = 2^d$ we introduce a transition that replaces an agent in 2^d by a multiset of agents r with $\text{sum}(r) = 2^d - m$, preserving the residue. Let $b_d b_{d-1}...b_0$ be the binary encoding of $2^d - m$, and let $\{i_1, ..., i_j\}$ be the positions such that $b_{i_1} = \cdots = b_{i_j} = 1$. The transition is:

$$2^d, 0, ..., 0 \quad \mapsto \quad 2^{i_1}, ..., 2^{i_j} \qquad \langle\text{modulo}\rangle$$

These transitions are enough, but we also add a transition that takes d agents in 2^d and replaces them by agents with sum $d \cdot 2^d$ mod m. Intuitively, this makes the protocol faster. Let $b_d b_{d-1}...b_0$ and $\{i_1, ..., i_j\}$ be as above, but for $d \cdot 2^d$ mod m instead of $2^d - m$.

$$2^d, ..., 2^d \quad \mapsto \quad 2^{i_1}, ..., 2^{i_j}, 0, ..., 0 \qquad \langle\text{fast modulo}\rangle$$

Helpers. We set $H := \wr 3d \cdot 0 \wr$, i.e. the computer initially places at least $3d$ helper agents in state 0. This makes sure one can always execute the next ⟨modulo⟩ or ⟨fast modulo⟩ transition: if no more agents can be combined, there are at most d agents in the states $2^0, ..., 2^{d-1}$. Thus, there are at least $2d$ agents in the states 0 and 2^d, enabling one of these transitions. Observe that for every initial configuration $C_I + C_H$ we have $\text{sum}(C_I + C_H) = \text{sum}(C_I)$, and so, abusing language, every initial configuration for C_I is also a representation of z.

Output function. The computer eventually reaches a support configuration with at most one agent in every state except for 0. Thus, for every support set $S \subseteq Q$, we define $O(S) := 1$ if $\text{sum}(S) \equiv_m c$, and $O(S) = 0$ else. We show the existence of a small boolean circuit for the output function O in the proof of Lemma 8; this can be found in Appendix B.1.

▶ **Lemma 8.** *Let* $\varphi := \sum_{i=1}^{v} a_i x_i \equiv_m c$*, where* $a_i \in \{2^{d-1}, ..., 2^1, 2^0\}$ *for every* $1 \leq i \leq v$ *and* $c \in \{0, ..., m-1\}$ *with* $d := \lceil \log_2 m \rceil$*. There is a bounded computer of size* $\mathcal{O}(d)$ *deciding* φ*.*

The left half of Figure 1 shows the population computer for $\varphi = (8x + 5y \equiv_{11} 4)$.

6.2 Population computers for threshold predicates

We sketch the construction of population computers $\mathcal{P}_\varphi$ for threshold predicates $\varphi := \sum_{i=1}^{v} a_i x_i \geq c$, where $a_i \in \{2^j, 2^{-j} \mid j \geq 0\}$ for every $1 \leq i \leq v$ and $c \in \mathbb{N}$. As the construction is similar to the construction for remainder, we will focus on the differences and refer to Appendix B.2 for details.

As for remainder, we work with representations that are multisets of powers of 2. However, they represent the sum of their elements (without modulo) and we allow both positive and negative powers of 2. Similar to the remainder construction, the computer transforms any representation into a *support representation* without changing the represented value. Then, the computer decides the predicate using only the support of that representation.

Again, there are ⟨combine⟩ transitions that allow agents with the same value to combine. Instead of modulo transitions, ⟨cancel⟩ transitions further simplify the representation: $2^i, -2^i \mapsto 0, 0$. Note that even after exhaustively applying ⟨combine⟩ and ⟨cancel⟩ there can still be many agents in 2^d or many agents in -2^d. This has two consequences:

- In the construction for general predicates of Section 6.3, we need that computers for remainder and threshold move most agents to state 0. In the remainder construction, all but a constant number of agents are moved to 0. In contrast, the threshold construction does not have this property. Thus, we do not design a single computer for a given threshold predicate φ but a family: one for every degree d larger than some minimum degree $d_0 \in \Omega(|\varphi|)$. Intuitively, larger degrees result in a larger fraction of agents in 0.
- Assume we detect agents in 2^d (-2^d is analogous). If there are many, the predicate is true. However, if there is just one, then the represented value might be small, due to negative contributions $-2^0, ..., -2^{d-1}$. We cannot distinguish the two cases, so we add transition ⟨cancel 2nd highest⟩: $2^d, -2^{d-1} \mapsto 2^{d-1}, 0$. It ensures that agents cannot be present in both 2^d and -2^{d-1}; therefore, an agent in 2^d certifies a value of at least 2^{d-1}.

The right half of Figure 1 shows the population computer for $\varphi = (-2x + y \geq 5)$ with degree $d = 4$. Appendix B.2 proves:

▶ **Lemma 9.** *Let* $\varphi := \sum_{i=1}^{v} a_i x_i \geq c$*, where* $a_i \in \{2^j, 2^{-j} \mid j \geq 0\}$ *for every* $1 \leq i \leq v$*. For every* $d \geq \max\{\lceil \log_2 c \rceil + 1, \lceil \log_2 |a_1| \rceil, ..., \lceil \log_2 |a_v| \rceil\}$ *there is a bounded computer of size* $\mathcal{O}(d)$ *that decides* φ*.*

6.3 Population computers for all Presburger predicates

We present a construction that, given threshold or remainder predicates $\varphi_1, ..., \varphi_s$, yields a population computer $\mathcal{P}$ deciding an arbitrary given boolean combination $B(\varphi_1, ..., \varphi_s)$ of $\varphi_1, ..., \varphi_s$. We only sketch the construction, see Appendix B.3 for details. We use the example $\varphi_1 = (y - 2x \geq 5)$, $\varphi_2 = (8x + 5y \equiv_{11} 4)$ and $B(\varphi_1, \varphi_2) = \varphi_1 \vee \varphi_2$. The result of the construction for this example is shown in Figure 1. The construction has 6 steps:

1. Rewrite Predicates. The constructions in Sections 6.1 and 6.2 only work for predicates where all coefficients are powers of 2. We transform each predicate φ_i into a new predicate φ'_i where all coefficients are decomposed into their powers of 2. In our example, $\varphi'_1 := \varphi_1$ because all coefficients are already powers of 2. However, $\varphi_2(x, y) = (8x + 5y \equiv_{11} 4)$ is rewritten as $\varphi'_2(x, y_1, y_2) := (8x + 4y_1 + 1y_2 \equiv_{11} 4)$ because $5 = 4 + 1$. Note that $\varphi_2(x, y) = \varphi'_2(x, y, y)$

holds for every $x, y \in \mathbb{N}$. Let r be the size of the largest split of a coefficient, i.e. $r = 2$ in the example.

2. Construct Subcomputers. For every $1 \leq i \leq s$, if φ_i is a remainder predicate, then let $\mathcal{P}_i$ be the computer defined in Section 6.1. If φ_i is a threshold predicate, then let $\mathcal{P}_i$ be the computer of Section 6.2, with $d = d_0 + \lceil \log_2 s \rceil$. We explain this choice of d in step 5.

3. Combine Subcomputers. Take the disjoint union of $\mathcal{P}_i$, but merging their 0 states. More precisely, rename all states $q \in Q_i$ to $(q)_i$, with the exception of state 0. Construct a computer with the union of all the renamed states and transitions. Figure 1 shows the Petri net representation of the computer so obtained for our example. We call the combined 0 state *reservoir* as it holds agents with no value that are needed for various tasks like input distribution.

4. Input Distribution. For each variable x_i add a corresponding new input state x_i. Then add a transition that takes an agent in state x_i and agents in 0 and distributes agents to the input states of the subcomputers that correspond to x_i. In our example, we add two states x and y and the transitions $x, 0 \mapsto (1)_1, (8)_2$ and $y, 0, 0 \mapsto (-2)_1, (4)_2, (1)_2$. The distribution for x needs one helper, because we need one agent in each subcomputer. The distribution for y needs two helpers, one for $\mathcal{P}_1$ and two for $\mathcal{P}_2$, as $5y$ was split into $4y_1 + 1y_2$. This way, once the input states are empty, the correct value is distributed to each subcomputer. Crucially, this input distribution can be fast as it is not reversible.

5. Add Extra Helpers. In addition to all helpers from the subcomputers, add $r - 1$ more helpers to state 0. Intuitively, this allows to distribute the first input agent. Because of our choice for d in threshold subcomputers, each subcomputer returns most agents back to state 0. More precisely, for each distribution the number of agents that do not get returned to 0 only increases by at most $\frac{1}{s}$ (per subcomputer). So in total only one agent is "consumed" per distribution and enough agents are returned to 0 for the next distribution to occur. In our example, the agents that stay in each of the $s = 2$ subcomputers only increases by at most $\frac{1}{2}$ per distribution. (In fact, remainder subcomputers return all distributed agents.)

6. Combine Output. Note that we can still decide φ_i from the support of the states in the corresponding subcomputer $\mathcal{P}_i$. We compute the output for φ by combining the outputs of the subcomputers $\mathcal{P}_1, ..., \mathcal{P}_s$ according to $B(\varphi_1, ..., \varphi_s)$. In our example, we set the output to 1 if and only if the output of $\mathcal{P}_1$ or $\mathcal{P}_2$ is 1.

In Appendix B.3, we show that this computer is succinct, correct and bounded:

▶ **Theorem 1.** *For every predicate $\varphi \in$ QFPA there exists a bounded population computer of size $\mathcal{O}(|\varphi|)$ that decides φ.*

7 Converting Population Computers to Population Protocols

In this section we prove Theorem 2. We proceed in four steps, which must be carried out in the given order. Section 7.1 converts any bounded computer $\mathcal{P}$ for double(φ) of size m into a *binary* bounded computer $\mathcal{P}_1$ with $\mathcal{O}(m^2)$ states. Section 7.2 converts $\mathcal{P}_1$ into a binary bounded computer $\mathcal{P}_2$ with a *marked consensus output function* (a notion defined in the section). Section 7.3 converts $\mathcal{P}_2$ into a binary bounded computer $\mathcal{P}_3$ for φ — not double(φ) — with a marked consensus output function *and no helpers*. Section 7.4 shows that $\mathcal{P}_3$ runs within $\mathcal{O}(n^3)$ interactions. Finally, we convert $\mathcal{P}_3$ to a binary terminating (not necessarily bounded) computer $\mathcal{P}_4$ with a *normal consensus output* and no helpers, also running within $\mathcal{O}(n^3)$ interactions. This uses standard ideas; for space reasons it is described only in the full version at Appendix F. Similarly, the other conversions and results are only sketched, with details in the appendix.

7.1 Removing multiway transitions

We transform a bounded population computer with k-way transitions $r \mapsto s$ such that $|\text{supp}(r)| \leq 2$ into a binary bounded population computer. Let us first explain why the construction introduced in [9, Lemma 3], which works for arbitrary transitions $r \mapsto s$, is too slow. In [9], the 3-way transition $t : q_1, q_2, q_3 \mapsto q_1', q_2', q_3'$ is simulated by the transitions

$$t_1 : q_1, q_2 \mapsto w, q_{12} \qquad t_2 : q_{12}, q_3 \mapsto c_{12}, q_3' \qquad t_3 : q_3', w \mapsto q_1', q_2' \qquad \bar{t}_1 : w, q_{12} \mapsto q_1, q_2$$

Intuitively, the occurrence of t_1 indicates that two agents in q_1 and q_2 want to execute t, and are waiting for an agent in q_3. If the agent arrives, then all three execute t_2t_3, which takes them to q_1', q_2', q_3'. Otherwise, the two agents must be able to return to q_1, q_2 to possibly execute other transitions. This is achieved by the "revert" transition $\bar{t}_1$. The construction for a k-way transition has "revert" transitions $\bar{t}_1, ..., \bar{t}_{k-2}$. As in Example 6 and Example 7, these transitions make the final protocol very slow.

We present a gadget without "revert" transitions that works for k-way transitions $r \mapsto s$ satisfying $|\text{supp}(r)| \leq 2$. Figure 2 illustrates it, using Petri net notation, for the 5-way transition $t : \wr 3p, 2q \wr \mapsto \wr a, b, c, d, e \wr$. In the gadget, states p and q are split into $(p, 0), ..., (p, 3)$

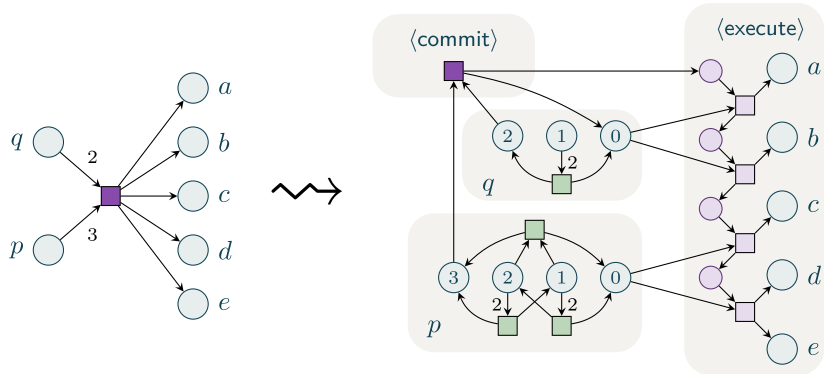

Figure 2 Simulating the 5-way transition $\wr 3 \cdot p, 2 \cdot q \mapsto a, b, c, d, e \wr$ by binary transitions.

and $(q, 0), ..., (q, 2)$. Intuitively, an agent in (q, i) acts as representative for a group of i agents in state q. Agents in $(p, 3)$ and $(q, 2)$ commit to executing t by executing the binary transition ⟨commit⟩. After committing, they move to the states $a, ..., e$ together with the other members of the group, who are "waiting" in the states $(p, 0)$ and $(q, 0)$. Note that ⟨commit⟩ is binary because of the restriction $|\text{supp}(r)| \leq 2$ for multiway transitions.

To ensure correctness of the conversion, agents can commit to transitions if they represent more than the required amount. In this case, the initiating agents would commit to a transition and then elect representatives for the superfluous agents, before executing the transition. This requires additional intermediate states.

Appendix C formalises the gadget and proves its correctness and speed.

7.2 Converting output functions to marked-consensus output functions

We convert a computer with an arbitrary output function into another one with a *marked-consensus* output function. An output function is a *marked-consensus* output function if there are disjoint sets of states $Q_0, Q_1 \subseteq Q$ such that $O(S) := b$ if $S \cap Q_b \neq \emptyset$ and $S \cap Q_{1-b} = \emptyset$, for $b \in \{0,1\}$, and $O(S) := \bot$ otherwise. Intuitively, for every $S \subseteq Q$ we have $O(S) = 1$ if all agents agree to avoid Q_0 (consensus), and at least one agent populates Q_1 (marked consensus). We only sketch the construction, a detailed description as well as a graphical example can be found in Appendix D.

Our starting point is some bounded and binary computer $\mathcal{P} = (Q, \delta, I, O, H)$, e.g. as constructed in Section 7.1. Let (G, E) be a boolean circuit with only NAND-gates computing the output function O. We simulate $\mathcal{P}$ by a computer $\mathcal{P}'$ with a marked consensus output and $\mathcal{O}(|Q| + |G|)$ states. This result allows us to bound the number of states of $\mathcal{P}'$ by applying well known results on the complexity of Boolean functions.

Intuitively, $\mathcal{P}'$ consists of two processes running asynchronously in parallel. The first one is (essentially, see below) the computer $\mathcal{P}$ itself. The second one is a gadget that simulates the execution of G on the support of the current configuration of $\mathcal{P}$. Whenever $\mathcal{P}$ executes a transition, it raises a flag indicating that the gadget must be reset (for this, we duplicate each state $q \in Q$ into two states $(q,+)$ and $(q,-)$, indicating whether the flag is raised or lowered). Crucially, $\mathcal{P}$ is bounded, and so it eventually performs a transition *for the last time*. This resets the gadget for the last time, after which the gadget simulates (G, E) on the support of the terminal configuration reached by $\mathcal{P}$.

The gadget is designed to be operated by one *state-helper* for each $q \in Q$, with set of states $Q_{\text{supp}}(q)$, and a *gate-helper* for each gate $g \in G$, with set of states $Q_{\text{gate}}(g)$, defined as follows:

- $Q_{\text{supp}}(q) := \{q\} \times \{0, 1, !\}$. These states indicate that q belongs/does not belong to the support of the current configuration (states $(q, 0)$ and $(q, 1)$), or that the output has changed from 0 to 1 (state $(q, !)$).
- $Q_{\text{gate}}(g) := \{g\} \times \{0, 1, \bot\}^3$ for each gate $g \in G$, storing the current values of the two inputs of the gate and its output. Uninitialised values are stored as $\bot$.

Recall that a population computer must also remain correct for a larger number of helpers. This is ensured by letting all helpers populating one of these sets, say $Q_{\text{supp}}(q)$, perform a leader election; whenever two helpers in states of $Q_{\text{supp}}(q)$ meet, one of them becomes a non-leader, and a flag requesting a complete reset of the gadget is raised. All resets are carried out by a *reset-helper* with set of states $Q_{\text{reset}} := \{0, ..., |Q| + |G|\}$, initially in state 0. (Reset-helpers also carry out their own leader election!) Whenever a reset is triggered, the reset-helper contacts all other $|Q| + |G|$ helpers in round-robin fashion, asking them to reset the computation.

Eventually the original protocol $\mathcal{P}$ has already reached a terminal configuration with some support Q_{term}, each set $Q_{\text{supp}}(q)$ and $Q_{\text{gate}}(g)$ is populated by exactly one helper, and all previous resets are terminated. From this moment on, $\mathcal{P}$ never changes its configuration. The $|Q|$ state-helpers detect the support Q_{term} of the terminal configuration by means of transitions that move them to the states $Q_{\text{term}} \times \{1\}$ and $(Q \setminus Q_{\text{term}}) \times \{0\}$; the gate-helpers execute (G, E) on input Q' by means of transitions that move them to the states describing the correct inputs and outputs for each gate. State-helpers use $Q \times \{!\}$ as intermediate states, indicating that the circuit must recompute its output.

It remains to choose the sets Q_0 and Q_1 of states the marked consensus output. We do it according to the output b of the output gate $g_{\text{out}} \in G$: Q_b is the set of states of $Q_{\text{gate}}(g_{\text{out}})$ corresponding to output b.

7.3 Removing helpers

We convert a bounded binary computer $\mathcal{P}$ deciding the predicate double(φ) over variables $x_1, ..., x_k, x'_1, ..., x'_k$ into a computer $\mathcal{P}'$ with no helpers deciding φ over variables $x_1, ..., x_k$. In [8], a protocol with helpers and set of states Q is converted into a protocol without helpers with states $Q \times Q$. We sketch a better construction that avoids the quadratic blowup. A detailed description can be found in Appendix E.

Let us give some intuition first. All agents of an initial configuration of $\mathcal{P}'$ are in input states. $\mathcal{P}'$ simulates $\mathcal{P}$ by *liberating* some of these agents and transforming them into helpers, without changing the output of the computation. For this, two agents in an input state x_i are allowed to interact, producing one agent in x'_i and one "liberated" agent, which can be used as a helper. This does not change the output of the computation, because double(φ)$(..., x_i, ..., x'_i, ...) =$ double(φ)$(..., x_i - 2, ..., x'_i + 1, ...)$ holds by definition of double(φ).

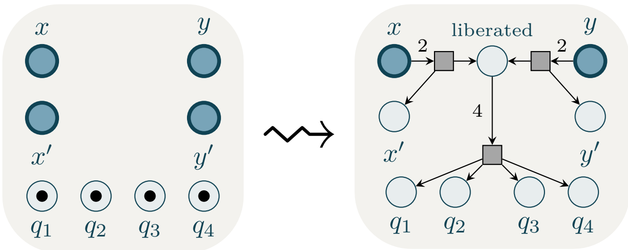

Figure 3 Illustration in graphical Petri net notation (see Section 3) of construction that removes helpers. Initial states are highlighted.

Figure 3 illustrates this idea. Assume $\mathcal{P}$ has input states x, y, x', y' and helpers $H = \wr q_1, q_2, q_3, q_4 \wr$, as shown on the left-hand side. Assume further that $\mathcal{P}$ computes a predicate double(φ)(x, y, x', y'). The computer $\mathcal{P}'$ is shown on the right of the figure. The additional transitions liberate agents, and send them to the helper states H. Observe that the initial states of $\mathcal{P}'$ are only x and y. Let us see why $\mathcal{P}'$ decides $\varphi(x, y)$. As the initial configuration of $\mathcal{P}'$ for an input x, y puts no agents in x', y', the computer $\mathcal{P}'$ produces the same output on input x, y as $\mathcal{P}$ on input $x, y, 0, 0$. Since $\mathcal{P}$ decides double(φ) and double(φ)$(x, y, 0, 0) = \varphi(x, y)$ by the definition of double(φ), we are done. We make some remarks:

- $\mathcal{P}'$ may liberate more agents than necessary to simulate the multiset H of helpers of $\mathcal{P}$. This is not an issue, because by definition additional helpers do not change the output of the computation.
- If the input is too small, $\mathcal{P}'$ cannot liberate enough agents to simulate H. Therefore, the new computer only works for inputs of size $\Omega(|H|) = \Omega(|\varphi|)$.
- Even if the input is large enough, $\mathcal{P}'$ might move agents out of input states before liberating enough helpers. However, the computers of Section 6 can only do this if there are enough helpers in the reservoir state (see point 3. in Section 6.3). Therefore, they always generate enough helpers when the input is large enough.

7.4 A $\mathcal{O}(n^3)$ bound on the expected interactions

We show that the computer obtained after the previous conversion runs within $\mathcal{O}(n^3)$ interactions. We sketch the main ideas; the details are in Appendix G.

We introduce *potential functions* that assign to every configuration a positive *potential*, with the property that executing any transition strictly decreases the potential. Intuitively, every transition "makes progress". We then prove two results: (1) under a mild condition, a computer has a potential function iff it is bounded, and (2) every binary computer with a potential function and no helpers, i.e. any bounded computer for which speed is defined, stabilises within $\mathcal{O}(n^3)$ interactions. This concludes the proof.

Fix a population computer $\mathcal{P} = (Q, \delta, I, O, H)$.

▶ **Definition 10.** *A function $\Phi : \mathbb{N}^Q \to \mathbb{N}$ is* linear *if there exist weights $w : Q \to \mathbb{N}$ s.t. $\Phi(C) = \sum_{q \in Q} w(q)C(q)$ for every $C \in \mathbb{N}^Q$. We write $\Phi(q)$ instead of $w(q)$. A* potential function *(for $\mathcal{P}$) is a linear function Φ such that $\Phi(r) \geq \Phi(s) + |r| - 1$ for all $(r \mapsto s) \in \delta$.*

Observe that k-way transitions reduce the potential by $k-1$, binary transitions by 1. At this point, we consider only binary computers, but this distinction becomes relevant for the refined speed analysis.

If a population computer has a potential function, then every run executes at most $\mathcal{O}(n)$ transitions, and so the computer is bounded. Applying Farkas' Lemma we can show that the converse holds for computers in which every state can be populated—a mild condition, since states that can never be populated can be deleted without changing the behaviour.

▶ **Lemma 11.** *If $\mathcal{P}$ has a reachable configuration C_q with $C_q(q) > 0$ for each $q \in Q$, then $\mathcal{P}$ is bounded iff there is a potential function for $\mathcal{P}$.*

Consider now a binary computer with a potential function and no helpers. At every non-terminal configuration, at least one (binary) transition is enabled. The probability that two agents chosen uniformly at random enable this transition is $\Omega(1/n^2)$, and so a transition occurs within $\mathcal{O}(n^2)$ interactions. Since the computer has a potential function, every run executes at most $\mathcal{O}(n)$ transitions, and so the computer stabilises within $\mathcal{O}(n^3)$ interactions.

The final step to produce a population protocol is to translate computers with marked-consensus output function into computers with standard consensus output function, while preserving the number of interactions. For space reasons this construction is presented in Appendix F.

8 Rapid Population Computers: Proving a $\mathcal{O}(n^2)$ Bound

We refine our running-time analysis to show that the population protocols we have constructed actually stabilise within $\mathcal{O}(n^2)$ interactions. We continue to use potential functions, as introduced in Section 7.4, but improve our analysis as follows:

- We introduce *rapidly-decreasing* potential functions. Intuitively, their existence shows that progress is not only *possible*, but also *likely*. We prove that they certify stabilisation within $\mathcal{O}(n^2)$ interactions.
- We introduce *rapid* population computers, as computers with rapidly-decreasing potential functions that also satisfy some technical conditions. We convert rapid computers into protocols with $O(|\varphi|)$ states, and show that the computers of Section 6 are rapid.

In order to define rapidly-decreasing potential functions, we need a notion of "probability to execute a transition" that generalises to multiway transitions and is preserved by our conversions. At a configuration C of a protocol, the probability of executing a binary transition $t = (p, q \mapsto p', q')$ is $C(q)C(p)/n(n-1)$. Intuitively, leaving out the normalisation factor $1/n(n-1)$, the transition has "speed" $C(q)C(p)$, proportional in the *product* of the number of agents in p and q. But for a multiway transition like $q, q, p \mapsto r_1, r_2, r_3$ the situation changes. If $C(q) = 2$, it does not matter how many agents are in p – the transition is always going to take $\Omega(n^2)$ interactions. We therefore define the speed of a transition as $\min\{C(q), C(p)\}^2$ instead of $C(q)C(p)$.

For the remainder of this section, let $\mathcal{P} = (Q, \delta, I, O, H)$ denote a population computer.

▶ **Definition 12.** *Given a configuration $C \in \mathbb{N}^Q$ and some transition $t = (r \mapsto s) \in \delta$, we let* $\text{tmin}_t(C) := \min\{C(q) : q \in \text{supp}(r)\}$*. For a set of transitions $T \subseteq \delta$, we define* $\text{speed}_T(C) := \sum_{t\in T} \text{tmin}_t(C)^2$*, and write* $\text{speed}(C) := \text{speed}_\delta(C)$ *for convenience.*

▶ **Definition 13.** *Let Φ denote a potential function for $\mathcal{P}$ and let $\alpha \geq 1$. We say that Φ is* α-rapidly decreasing *at a configuration C if* $\text{speed}(C) \geq (\Phi(C) - \Phi(C_{\text{term}}))^2/\alpha$ *for all terminal configurations C_{term} with $C \to C_{\text{term}}$.*

We have not been able to find potential functions for the computers of Section 6 that are rapidly decreasing at every reachable configuration, only at reachable configurations with sufficiently many helpers, defined below. Fortunately, that is enough for our purposes.

▶ **Definition 14.** *$C \in \mathbb{N}^Q$ is* well-initialised *if C is reachable and $C(I) + |H| \leq \frac{2}{3}n$.*

Observe that an initial configuration C can only be well-initialised if $C(\text{supp}(H)) \in \Omega(C(I))$. We now define *rapid* population computers, and state the result of our improved analysis.

▶ **Definition 15.** *$\mathcal{P}$ is* α-rapid *if*

1. *it has a potential function Φ which is α-rapidly decreasing in well-initialised configurations,*
2. *every state of $\mathcal{P}$ but one has at most 2 outgoing transitions,*
3. *all configurations in $\mathbb{N}^I$ are terminal, and*
4. *for all transitions $t = (r \mapsto s)$, $q \in I$ we have $r(q) \leq 1$ and $s(q) = 0$.*

▶ **Theorem 3.** **(a)** *The population computers constructed in Theorem 1 are $\mathcal{O}(|\varphi|^3)$-rapid.*

(b) *Every α-rapid population computer of size m deciding* $\text{double}(\varphi)$ *can be converted into a terminating population protocol with $\mathcal{O}(m)$ states that decides φ in $\mathcal{O}(\alpha\, m^4 n^2)$ interactions for inputs of size $\Omega(m)$.*

The detailed proofs can be found in the Appendix, in the following sections. The proof of (a) is given in Appendix B. For (b), we prove separate theorems for each conversion in Appendices C, D, E, and F. To achieve a tighter analysis of our conversions, we generalise the notion of potential function; this is described in Appendix H.

9 Conclusions

We have shown that every predicate φ of quantifier-free Presburger arithmetic has a population protocol with $\mathcal{O}(\text{poly}(|\varphi|))$ states and $\mathcal{O}(|\varphi|^7 \cdot n^2)$ expected number of interactions. If only inputs of size $\Omega(|\varphi|)$ matter, we give a protocol with $\mathcal{O}(|\varphi|)$ states and the same speed. The obvious point for further improvement is the $|\varphi|^7$ factor in the expected number of interactions.

Our construction is close to optimal. Indeed, for every construction there is an infinite family of predicates for which it yields protocols with $\Omega(|\varphi|^{1/4})$ states [9]; further, it is known that every protocol for the majority predicate requires in $\Omega(n^2)$ interactions.

Acknowledgements. We thank the anonymous reviewers for many helpful remarks. In particular, one remark led to Lemma 11, which in turn led to a nicer formulation of Theorem 2, one of our main results.

A Proofs of Section 5: Previous constructions

A.1 The protocol of Example 6 is slow

Given a configuration C with $i < 2^d$ positive agents, the next configuration C' can have $i-1$, i, or $i+1$ positive agents. Let us give a lower bound on the probability of the case $i+1$, and an upper bound on the probability of the case $i-1$.

- $i \to i+1$. This happens whenever a non-zero agent with wealth different from 1 or -1 meets a zero agent. Since i is the number of positive agent, the configuration C has $n - i > n - 2^d$ zero agents. Further, since the total wealth is 2^d and there are less than 2^d non-zero agents, at least one agent has wealth bigger than 1. So the probability is at least $p^+ := 2(n-2^d)/n(n-1)$, and so $\Omega(1/n)$.
- $i \to i-1$. This can only happen if two non-zero agents meet. Since there are less than 2^d non-zero agents, $p^- := 2^d(2^d-1)/n(n-1)$ is an upper bound, and so the probability is $\mathcal{O}(1/n^2)$ for fixed d.

We analyse the following random walk with states is $Q = \{2^d, ..., 1\}$, with initial state 2^d, target 1, and probabilities p^+, p^-, and $1 - p^+ - p^- \in \mathcal{O}(1)$ of moving towards 2^d, 1, and staying in the same state.

For $i \in \{2^d, ..., 1\}$, let T_i be the first time to hit 1 starting in i. Since the induced stochastic process $X_0, X_1, ...$ is a Markov chain, we can use the strong Markov property, i.e. we can restart the process at any stopping time. We choose $T := \inf\{i \in \mathbb{N} : X_i \neq X_{i-1}\}$. If $i \in \{2^d-1, ..., 2\}$, then $X_T = i-1$ or $X_T = i+1$, with probability $p_1 \in \mathcal{O}(1/n)$ of being in $i-1$ and probability $p_2 = 1 - p_1 \in \Theta(1)$ of being in $i+1$. Importantly, these probabilities of moving left or right again do not depend on the location $2^d > i > 1$ we started at. If we started at 2^d, then $X_T = 2^d - 1$. We have $T_i = T + T_{X_T}$ and therefore $E[T_i] = p_1 E[T_{i-1}] + p_2 E[T_{i+1}]$. For the cases $i = 2^d$ and $i = 1$ we obtain $E[T_1] = 0$ and $E[T_{2^d}] = E[T] + E[T_{2^d-1}]$. In order to obtain a rudimentary lower bound, we use $E[T] \geq 1$ to obtain the inequalities

$$\begin{aligned} E[T_i] &\geq p_1 E[T_{i+1}] + p_2 E[T_{i-1}] + 1 && \text{for all } 2^d > i > 1 \\ E[T_{2^d}] &\geq E[T_{2^d-1}] + 1 \end{aligned}$$

which can be easily solved, yielding $E[T_{2^d}] \in (p_2/p_1))^{2^d d2} \in \Omega(n^{2^d-2})$. In fact, this is a well-known fact for a random walk biased in the wrong direction by a factor of p_2/p_1, which yields the same inequalities.

A.2 The toy protocol of Example 7 is slow

Recall that the toy protocol has states q_1, q_2 and transitions $q_1, q_1 \mapsto q_2, q_2$ and $q_1, q_2 \mapsto q_1, q_1$. The initial configuration C_0 puts n agents in q_1 and 0 agents in q_2.

Let $(C_0, C_1, ...)$ be the stochastic process induced by the toy protocol, where C_i indicates the configuration after i interactions. Since at every step agents are chosen independently and uniformaly at random, the process is a Markov chain. We can identify the state space of the chain with the set $\{0, 1, 2, ..., n\}$ via the mapping $C_t \mapsto C_t(q_1)$. At state i, three transitions can happen, leading to states $i+1$, i and $i-2$. The probabilities of moving to $i+1$ and $i-2$ are $\frac{i(n-i)}{n(n-1)}$ and $\frac{i(i-1)}{n(n-1)}$, respectively. The goal is to reach the state 0 from the state n.

In order to obtain a lower bound on the number of steps, let us reduce the states to $\{0, ..., \lfloor n/5 \rfloor\}$, replacing the transition $\lfloor n/5 \rfloor \mapsto \lfloor n/5 \rfloor + 1$ by a self-loop at $\lfloor n/5 \rfloor$, and starting at state $\lfloor n/5 \rfloor$ instead of n. This only reduces the number of steps to the goal. In this new chain, the quotient of the probabilities of moving to $i+1$ and $i-2$ is $\frac{i(i-1)}{i(n-i)} \leq \frac{i}{n-i} \leq \frac{1}{4}$

for all states i such that $i + 1$ and $i - 2$ exist. Using the same idea as in the analysis of Example 6, and choosing again the stopping time $T := \inf\{j \in \mathbb{N} : C_j \neq C_{j-1}\}$, we can simplify the chain further, without increasing the number of steps to the goal, into a chain with probability $4/5$ and $1/5$ of moving to from i to $i + 1$ and to $i - 2$, respectively.

The expected number of steps to the goal in this chain is the same as for a random walk with states $\{0, ..., \lfloor n/10 \rfloor\}$, biased by a factor of 2 in the wrong direction. Indeed, the fact that in the chain we move from i to $i - 2$, while in the random walk we move from i to $i - 1$, is compensated by the probability in the chain being lower by a factor of 4. This biased random walk needs $\Omega(2^{n/10})$ steps until it reaches 0 from $\lfloor n/10 \rfloor$.

B Population Computers for Presburger Predicates: Detailed Construction and Analysis

In this section we will give a detailed description of the constructions in Section 6, prove their correctness, and show that the resulting population computers are fast.

We show two two results; these encapsulate all properties used by other sections. First, we prove the existence of succinct and bounded population computers.

▶ **Theorem 1.** *For every predicate $\varphi \in QFPA$ there exists a bounded population computer of size $\mathcal{O}(|\varphi|)$ that decides φ.*

The proof is split over multiple sections. Appendix B.1 and Appendix B.2 introduce our construction for threshold and modulo predicates, and Appendix B.3 generalises this to boolean combinations of such predicates. Finally, Appendix B.4 shows that the construction has the desired size and is correct and bounded.

Second, we show that the resulting computers are rapid. The definition of rapid population computers can be found in Section 8.

▶ **Theorem 3a.** *The population computers constructed in Theorem 1 are $\mathcal{O}(|\varphi|^3)$-rapid.*

The technical properties are shown in Lemma 16, while the existence of a rapidly-decreasing potential function is shown as Lemma 21.

Notation

We often need to split an integer into its powers of two. Thus, we introduce the function $\text{bin}(x)$ that maps to an integer x the multiset that contains the (possibly negative) powers of 2 corresponding to binary representation of x, i.e. $\text{bin}(-13) = \wr -2^3, -2^2, -2^0 \wr$ and $\text{bin}(10) = \wr 2^3, 2^1 \wr$.

$$\text{sign}(x) := \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

$$\text{bin}(x) := \wr \text{sign}(x) \cdot 2^i \mid i\text{-th bit in the binary encoding of } |x| \text{ is } 1 \wr$$

Note that $\text{sum}(\text{bin}(x)) = x$ for all $x \in \mathbb{Z}$.

B.1 Remainder predicates

Let $Pow^+ = \{2^i \mid i \geq 0\}$ be the set of positive powers of 2. Let φ be a remainder predicates of the form $\sum_{i=1}^{v} a_i x_i \equiv_m c$, where $a_i \in Pow^+ \cap \{0, ..., m-1\}$ for every $1 \leq i \leq v$, $m \in \mathbb{N}$, and $c \in \{0, ..., m-1\}$. We call a multiset $r \in \mathbb{N}^{Pow^+}$ a representation.

Let $\text{rep}(r) := \text{sum}(r) \bmod m$ denote the remainder represented by r. For example, if $m = 11$ then $r_{18} := \wr 2^3, 2^3, 2^1 \wr$ represents $\text{rep}(r_{18}) = 7$. A *representation of degree d* only contains elements of $Pow_d^+ := \{2^d, 2^{d-1}, ..., 2^0\}$. A representation r is a *support representation* if $r(x) \leq 1$ for every $x \in Pow^+$, i.e. if $\text{rep}(r)$ is completely determined by its support. For example, r_{18} is not a support representation of 7, but $\wr 2^5, 2^3 \wr$ and $\wr 2^2, 2^1, 2^0 \wr$ are.

Let $d := \lceil \log_2 m \rceil$ be the degree. Now we define the computer $\mathcal{P}_\varphi := (Q, \delta, I, O, H)$:

$$Q := Pow_d^+ \cup \{0\} \qquad I := \{a_1, ..., a_v\}$$

$$O(S \subseteq Q) := \begin{cases} 1 & \text{if } z(\wr q \mid q \in S \wr) = c \\ 0 & \text{otherwise} \end{cases} \qquad H := \wr (3d) \cdot 0 \wr$$

We now define the transitions in δ.

$$\begin{array}{llll} 2^i, 2^i & \mapsto & 0, 2^{i+1} & \text{for } 0 \le i \le d-1 \quad \langle\textsf{combine}\rangle \\ 2^d, 0, ..., 0 & \mapsto & \wr 0, 0 \wr + \text{bin}(2^d - m) & \langle\textsf{modulo}\rangle \\ \wr d \cdot 2^d \wr & \mapsto & \text{bin}(d2^d \bmod m) + \wr 0, ..., 0 \wr & \langle\textsf{fast modulo}\rangle \end{array}$$

▶ **Lemma 8.** *Let $\varphi := \sum_{i=1}^{v} a_i x_i \equiv_m c$, where $a_i \in \{2^{d-1}, ..., 2^1, 2^0\}$ for every $1 \le i \le v$ and $c \in \{0, ..., m-1\}$ with $d := \lceil \log_2 m \rceil$. There is a bounded computer of size $\mathcal{O}(d)$ deciding φ.*

Proof. The sum of all agents never increases and cannot become negative. Initially, each agent has a value of less than m and ⟨modulo⟩ / ⟨fast modulo⟩ transitions reduce the sum by at least m. Thus, there can be at most n ⟨modulo⟩ / ⟨fast modulo⟩ transitions. Between tween two ⟨modulo⟩ / ⟨fast modulo⟩ transitions, there can be at most n ⟨combine⟩ transitions, because they increase the number of agents by one. This implies that $\mathcal{P}_\varphi$ is bounded.

After at most $n(n+1)$ transitions, we reach a terminal configuration C. We know that $C(2^i) < 1$ for $0 \le i < d$, because all ⟨combine⟩ transitions are disabled in C. Thus, $C(0) + C(2^d) \ge 2d$, because we started with $3d$ helpers. We know $C(2^d) = 0$, because otherwise either ⟨modulo⟩ or ⟨fast modulo⟩ would be enabled. Therefore C is a support configuration. The value of rep is invariant throughout the configurations of a run. Thus, we can decide the predicate by evaluating $\text{rep}(C) = c$. Because of the choice for d, we know $\text{rep}(C) = \mathit{sum}(C)$ or $\text{rep}(C) = \mathit{sum}(C) - m$. Therefore, $\text{rep}(C) = c$ can be checked by a boolean circuit of size $\mathcal{O}(d)$. There are d ⟨combine⟩ transitions with size 2 and we know $|\langle\textsf{modulo}\rangle| \le d+2$ and $|\langle\textsf{fast modulo}\rangle| \le d$. Thus, $\text{size}(\mathcal{P}_\varphi) := |Q| + |H| + \text{size}(O) + \sum_{t\in\delta}|t| = (d+2) + 3d + \mathcal{O}(d) + \sum_{t\in\delta}|t| = \mathcal{O}(d)$. ◀

B.2 Threshold predicates

Let $\mathit{Pow} = \{2^i, -2^i \mid i \ge 0\}$ be the set of positive and negative powers of 2. Let φ be a threshold predicates of the form $\sum_{i=1}^{v} a_i x_i \ge c$, where $a_i \in \mathit{Pow}$ for every $1 \le i \le v$ and $c \in \mathbb{N}$. We call a multiset r of powers of 2 a representation and let $z(r) := \text{sum}(r)$ denote the integer represented by r. For example, $r_{11} := \wr 2^3, 2^3, 2^0, -2^2, -2^1 \wr$ represents $z(r_{11}) = 11$. A *representation of degree d* only contains elements of $\mathit{Pow}_d := \{2^d, 2^{d-1}, ..., 2^0, -2^0, ..., -2^{d-1}, -2^d\}$. For example, r_{11} is a representation of degree 3.

Let $d_0 := \max\{\lceil \log_2 c \rceil + 1, \lceil \log_2 |a_1| \rceil, ..., \lceil \log_2 |a_v| \rceil\}$ be the minimum degree. Now we define the computer $\mathcal{P} := (Q, \delta, I, O, H)$ for degree $d \ge d_0$:

$$Q := \mathit{Pow}_d \cup \{0\} \qquad I := \{a_1, ..., a_v\}$$

$$O(S \subseteq Q) := \begin{cases} 1 & \text{if } z\big(\wr q \mid q \in S \wr\big) \ge c \\ 0 & \text{otherwise} \end{cases} \qquad H := \wr d \cdot 0 \wr$$

Note that the computer is still bounded and correct, even if we do not add helpers. However, they make sure that *all* input agents are returned to state 0. This property is used in the general construction of Section 6.3 / Appendix B.3. The transitions in δ are, with $i \in [0, d-1]$:

$$\begin{array}{llllll} -2^i, 2^i & \mapsto & 0, 0 & -2^d, 2^d & \mapsto & 0, 0 \quad \langle\textsf{cancel}\rangle \\ 2^i, 2^i & \mapsto & 0, 2^{i+1} & -2^i, -2^i & \mapsto & 0, -2^{i+1} \quad \langle\textsf{combine}\rangle \\ 2^d, -2^{d-1} & \mapsto & 0, 2^{d-1} & -2^d, 2^{d-1} & \mapsto & 0, -2^{d-1} \quad \langle\textsf{cancel 2nd highest}\rangle \end{array}$$

▶ **Lemma 9.** *Let* $\varphi := \sum_{i=1}^{v} a_i x_i \geq c$*, where* $a_i \in \{2^j, 2^{-j} \mid j \geq 0\}$ *for every* $1 \leq i \leq v$*. For every* $d \geq \max\{\lceil \log_2 c \rceil + 1, \lceil \log_2 |a_1| \rceil, ..., \lceil \log_2 |a_v| \rceil\}$ *there is a bounded computer of size* $\mathcal{O}(d)$ *that decides* φ*.*

Proof. $\mathcal{P}_\varphi$ is bounded because each transition increases the number of agents in state 0. Thus, after at most n transitions, we reach a terminal configuration C. No transition changes the represented value. Thus, we can decide the predicate by comparing $z(C)$ with the threshold c.

We will now describe an algorithm that decides $z(C) \geq c$ from the support of C. First note that as ⟨cancel⟩ and ⟨cancel⟩ are not enabled, we know:

$$C(x) + C(-x) \leq 1 \qquad \text{for every } x \in Pow_{d-1} \qquad (1)$$

There are three cases:

- $C(2^d) > 0$. Because ⟨cancel 2nd highest⟩ is disabled, we know $C(-2^{d-1}) = 0$. Then, because of (1), we know that the value of all other agents must be larger then -2^{d-1}. Thus, $z(C) > 2^d - 2^{d-1} = 2^{d-1} \geq |c|$ and we directly know $z(C) \geq c$.
- $C(-2^d) > 0$. Symmetric.
- $C(2^d) = 0 = C(-2^d)$. Because of (1), we can calculate $z(C)$ by adding up the values of all states in the support of C. We add the values in decreasing order of absolute value (i.e. $2^{d-1}, -2^{d-1}, 2^{d-2}, ..., 2^1, -2^1, 2^0, -2^0$). This way, the absolute value of all agents that have not been observed is getting smaller. Let x be the current sum after considering the states 2^i and -2^i. Using (1), all smaller powers of two can only change the total sum by at most $2^i - 1$. Thus, if $x \geq c + 2^i - 1$ we can directly output 1 and if $x < c - 2^i + 1$ we can directly output 0; i.e. we only need to continue if x is in the interval $[c - (2^i - 1), c + (2^i - 1) - 1]$. The size of the interval is $2 \cdot 2^i - 3$ and x is a multiple of 2^i. Thus, there are at most two values of x for which we need to continue.

This gives rise to a decision diagram with at most $2d + 2$ decisions that can be converted to a circuit of size $\mathcal{O}(d)$. Note that $\mathcal{P}_\varphi$ is binary and has $\mathcal{O}(d)$ transitions. Thus, $\text{size}(\mathcal{P}_\varphi) := |Q| + |H| + \text{size}(O) + \sum_{t \in \delta} |t| = (2d + 3) + d + \mathcal{O}(d) + \mathcal{O}(d) = \mathcal{O}(d)$. ◀

B.3 Construction for general predicates

Let φ be a positive boolean combination $B(\varphi_1, ..., \varphi_s)$ of pairwise different threshold or remainder predicates $\varphi_1, ..., \varphi_s$ over variables $X := \{x_1, ..., x_v\}$. Let a_i^j be the coefficient of variable $x_i \in X$ in predicate φ_j. Let $a_{\max}^j := \max_{i=1}^{v} |a_i^j|$. If φ_j is a remainder predicate, let m_j be the modulo parameter and c_j be the threshold parameter, i.e. $\varphi_j = \left(\sum_{i=1}^{v} a_i^j x_i \equiv_{m_j} c_j\right)$. Wlog we can assume that $0 \leq a_i^j < m_j$ and $0 \leq c_j < m_j$. If φ_j is a threshold predicate, let c_j denote the threshold parameter, i.e. $\varphi_j = \left(\sum_{i=1}^{v} a_i^j x_i \geq c_j\right)$.

Let $r := \max_{i=1}^{v} \sum_{j=1}^{s} \left|\text{bin}(a_i^j)\right|$ denote the size of the largest split. We now rewrite each predicate φ_j:

$$\varphi'_j := \begin{cases} \sum_{i=1}^{v} \sum_{e \in \text{supp}(\text{bin}(a_i^j))} e \cdot x_{i,e} \geq c_j & \text{if } \varphi_j \text{ is a threshold predicate} \\ \sum_{i=1}^{v} \sum_{e \in \text{supp}(\text{bin}(a_i^j))} e \cdot x_{i,e} \equiv_{m_j} c_j & \text{if } \varphi_j \text{ is a remainder predicate} \end{cases}$$

Then we construct the population computer $\mathcal{P}_j = (Q_j, \delta_j, I_j, O_j, H_j)$ for each φ'_j. If φ'_j is a remainder predicate, we follow the construction in Section 6.1 / Appendix B.1. In particular, the degree of $\mathcal{P}_j$ is $d_j := \lceil \log_2 m_j \rceil$. Otherwise, if φ'_j is a threshold predicate, then we choose the degree $d_j := \max\{\lceil \log_2 c_j \rceil + 1, \lceil \log_2 (s \cdot a_{\max}^j) \rceil\} + 4$ and follow the construction in

Section 6.2 / Appendix B.2. We remark that the addition of 4 is not necessary for correctness, but we will later use it to show that the protocol is fast.

For our states, we use $Q := X \cup \{(q)_j : q \in Q_j, q \neq 0\} \cup \{0\}$, and define the corresponding mappings renaming the states $q \in Q_j$.

$$\nu_j : Q_j \to Q, \quad q \mapsto \begin{cases} 0 & \text{if } q = 0 \\ (q)_j & \text{otherwise} \end{cases}$$

We define a new transition which distributes the input agents, for input $i \in \{1, ..., v\}$.

$$x_i, 0, ..., 0 \mapsto \sum_{j=1}^{s} \nu_j \circ \text{bin}(a_i^j) \qquad \langle\textsf{distribute}\rangle$$

Here, $\nu_j \circ M$ for a multiset $M \in \mathbb{N}^{Q_j}$ is the result of renaming the agents in M according to ν_j. We have to be careful if $|\sum_j \text{bin}(a_i^j)| \leq 1$, i.e. if an input agent should be distributed to 0 or 1 states. In that case the ⟨distribute⟩ transition is an interaction between less than two agents. We make these transitions binary by adding 0 agents that don't change their state.

In addition to ⟨distribute⟩, we also add the transitions of δ_j renamed accordingly, for $j = 1, ..., s$.

$$\nu_j \circ p \mapsto \nu_j \circ r \qquad \text{for } (p \mapsto r) \in \delta_j \qquad \langle\textsf{subcomputer}\rangle$$

Finally, we define the inputs $I := X$, the output function $O := B(O_1, ..., O_s)$, and the helpers $H := (\max(r, 2) - 1) \cdot \wr 0 \wr + \sum_{j=1}^{s} H_j$.

B.4 Size and correctness

In this section, we want to show that our constructions fulfils the claimed properties, except for speed (which we show separately in the next section, as it is more involved). The main focus is on showing the following result.

▶ **Theorem 1.** *For every predicate $\varphi \in$ QFPA there exists a bounded population computer of size $\mathcal{O}(|\varphi|)$ that decides φ.*

Before we start proving Theorem 1, we start by noting a few technical properties, which are needed by the conversions later on.

We use the notation of the previous section. In particular, we will write $\mathcal{P} = (Q, \delta, I, O, H)$ for the result of the conversion.

▶ **Lemma 16.** *No state in I has incoming transitions, all configurations in $\mathbb{N}^I$ are terminal, every state but one has at most 2 outgoing transitions, and $r(q) \leq 1$ for $q \in I$ and $(r \mapsto p) \in \delta$.*

Proof. These properties follow directly from the construction. In particular, the only outgoing transition of states in I is ⟨distribute⟩, which uses an agent in $0 \notin I$. Additionally, the only state with more than two outgoing transitions is 0. ◀

We will now proceed with the proof of Theorem 1, which will take up the remainder of this section. The following lemma argues that summing the sizes of the subprotocols is still linearly bounded in φ.

▶ **Lemma 17.** $\sum_{j=1}^{s} d_j \in \mathcal{O}(|\varphi|)$

Proof. We start by showing $|\varphi| \in \Theta(s \log_2 s)$.

Let $l := \max\{|\varphi_1|, ..., |\varphi_s|\}$ be the size of the largest predicate. Remember that the construction of $\mathcal{P}$ assumed that all threshold and remainder predicates are different. Because predicates are encoded in binary, there are $2^{l+1} - 1$ different predicates with a length of at most l. Thus, $2^{l+1} - 1 \geq s$ and therefore $l \geq \log_2(s+1) - 1$. The combined length of all predicates with length at most l is $\sum_{i=0}^{l} i2^i$. The length of φ must be at least as long as all predicates with size at most $l-1$ and thus:

$$\begin{aligned}
|\varphi| &\geq \sum_{i=0}^{l-1} i2^i \\
&\geq \sum_{i=0}^{\log_2(s+1)-2} i2^i \\
&= (\log_2(s+1) - 2) \cdot (2^{\log_2(s+1)-1+1} - 1) \\
&= (\log_2(s+1) - 2) \cdot s \\
&\geq s \log_2 s - 2s \\
&= \Theta(s \log_2 s)
\end{aligned}$$

Using this fact we know:

$$\begin{aligned}
\sum_{j=1}^{s} d_j &= \sum_{j=1}^{s} \mathcal{O}(\log_2 c_j + \log_2 s + \log_2 a^j_{\max}) && \text{(by def. of } d_j\text{)} \\
&= \mathcal{O}(s \log_2 s) + \sum_{j=1}^{s} \mathcal{O}(\log_2 c_j + \log_2 a^j_{\max}) \\
&= \mathcal{O}(s \log_2 s) + \mathcal{O}(|\varphi|) && \text{(by def. of } |\varphi|\text{)} \\
&= \mathcal{O}(|\varphi|)
\end{aligned}$$

◀

We can use this to show a bound on the size of the protocol.

▶ **Lemma 18.** $\text{size}(\mathcal{P}) \in \mathcal{O}(|\varphi|)$

Proof. This lets us prove that $\mathcal{P}$ is succinct:

$$\begin{aligned}
\text{size}(\mathcal{P}) &= |Q| + |H| + \text{size}(O) + \sum_{t \in \delta} |t| && \text{(by def. of } Q, H, \delta \text{ and } O\text{)} \\
&= |X| + \max(r, 2) - 1 + \mathcal{O}(s) + \sum_{t \in \delta_{\text{dist}}} |t| + \sum_{j=1}^{s} \text{size}(\mathcal{P}_j) \\
& && \text{(by definition of } \langle \textsf{distribute} \rangle \text{ and } |\varphi|\text{)} \\
&= \mathcal{O}(|\varphi|) + \sum_{j=1}^{s} \text{size}(\mathcal{P}_j) && \text{(by Lemmas 8 \& 9)} \\
&= \mathcal{O}(|\varphi|) + \sum_{j=1}^{s} \mathcal{O}(d_j) \\
&= \mathcal{O}(|\varphi|)
\end{aligned}$$

◀

Next, we show boundedness, which follows more or less directly from the boundedness of the subprotocols.

▶ **Lemma 19.** *$\mathcal{P}$ is bounded.*

Proof. $\mathcal{P}$ is constructed by distributing input agents to the subcomputers $\mathcal{P}_1, ..., \mathcal{P}_s$. By Lemma 8 and Lemma 9, each subcomputer $\mathcal{P}_j$ is bounded. Transition $\langle\mathsf{distribute}\rangle$ reduces the number of agents in the input states I. As this number cannot increase via other kinds of transitions, so $\langle\mathsf{distribute}\rangle$ executes only finitely often.

Afterwards, we can use Lemmata 8 and 9, which show that the subcomputers are bounded. Here, we note that $\langle\mathsf{distribute}\rangle$ only adds agents to input states of a subcomputer $\mathcal{P}_j$, so any transition sequence executed by the subcomputer $\mathcal{P}_j$ could also be executed by $\mathcal{P}_j$ when started in an initial configuration. ◀

Finally, we show the correctness of $\mathcal{P}$, concluding the proof of Theorem 1.

▶ **Lemma 20.** *$\mathcal{P}$ decides φ.*

Proof. Assume that in each terminal configuration C, there are no agents in input states, i.e. $C(X) = 0$. From the definition of δ_{dist} we know the distribution of an agent in state x_i adds the value a_i^j to each subcomputer $\mathcal{P}_j$. Further, it is easy to see that the construction of $\mathcal{P}$ keeps all invariants of the subcomputers intact. Because in each terminal configuration, all input agents where distributed, each subcomputer $\mathcal{P}_j$ still decides φ_j. Using the definition of the output function, we conclude that $\mathcal{P}$ decides φ.

We still need to show that $C(X) = 0$ in every terminal configuration C. Let x be the number of distributed input agents in C. We will now proof that $C(Q_j \setminus \{0\}) \leq H_j(0) + \frac{x}{s}$ for each subcomputer $\mathcal{P}_j$. Intuitively, we show that the number of agents in the non-zero states of $\mathcal{P}_j$ is at most the number of helpers of $\mathcal{P}_j$ plus a *"fair share"* (i.e. $\frac{1}{s}$) of all distributed agents. If $\mathcal{P}_j$ is a remainder predicate, then $C(Q_j \setminus \{0\}) \leq 3d_j = H_j(0)$ because otherwise a transition $t \in \delta_j$ would be enabled (see proof of Lemma 8). Intuitively, remainder predicates return 100% of all distributed agents. If $\mathcal{P}_j$ is a threshold predicate, we know $C\left(Q_j \setminus \left\{\left(2^{d_j}\right)_j, \left(-2^{d_j}\right)_j, 0\right\}\right) \leq d_j = H_j(0)$ and $C\left(\left(2^{d_j}\right)_j\right) = 0$ or $C\left(\left(-2^{d_j}\right)_j\right) = 0$, because the transitions $t \in \delta_j$ are disabled in C. Each of the x distribution transitions increased the absolute value of all agents in the states $Q_j \setminus \{0\}$ by at most a^j_{max}. Thus, the total absolute value in states Q_j is at most xa^j_{max}. Using the definition of d_j, this implies that $C(\left(2^{d_j}\right)_j) + C(\left(-2^{d_j}\right)_j) \leq \frac{xa^j_{\text{max}}}{2^{d_j}} \leq \frac{x}{s}$.

We started with at least $\max(r, 2){-}1 + \sum_{j=1}^{s} H_j(0)$ helpers in state 0 and distributed x input agents. Therefore:

$$
\begin{aligned}
C(0) &= C(Q \setminus X) - \sum_{j=1}^{s} C(Q_j \setminus \{0\}) \\
&\geq C(Q \setminus X) - \sum_{j=1}^{s} \left(H_j(0) + \frac{x}{s}\right) \\
&\geq \max(r, 2){-}1 + x + \sum_{j=1}^{s} H_j(0) - \sum_{j=1}^{s} \left(H_j(0) + \frac{x}{s}\right) \\
&= \max(r, 2){-}1
\end{aligned}
$$

Using the definition of r and the definition of δ_{dist}, we conclude $C(X) = 0$ as otherwise, one of the distribution transitions would be enabled in the terminal configuration C. ◀

B.5 Speed

We now move on to show that our construction is fast, i.e. that a rapidly-decreasing potential function for $\mathcal{P}$ exists. In particular, the goal is to prove the following lemma:

▶ **Lemma 21.** *Φ is $\mathcal{O}(|\varphi|^3)$-rapidly decreasing in all well-initialised configurations.*

The analysis will proceed via potential functions. While we already know that a potential function exists — $\mathcal{P}$ is bounded, so Lemma 11 applies — we will construct one explicitly to optimise the dependence on $|\varphi|$ in our analysis. Afterwards, we show that this potential function is rapidly-decreasing.

Potential Function

To give a potential function for our construction (specifically the computers produced by Theorem 1), the most interesting part are the subcomputers of remainder predicates, which we now examine in detail.

▶ **Example 22.** Let $\mathcal{P}$ refer to a subcomputer for a remainder predicate, with states $Q = \{0, 2^0, 2^1, ..., 2^d\}$, and let $d' := d - \lceil \log_2 6d \rceil$. We define a potential function Φ as follows, with $i \in 0, ..., d' - 1$ and $j \in \{0, ..., d - d'\}$.

$$\Phi(0) := 0 \qquad\qquad \Phi(2^i) := 2 \qquad\qquad \Phi(2^{d'+j}) := 2^j + 1$$

▶ **Lemma 23.** *Φ is a potential function for $\mathcal{P}$.*

Proof. We need to check that every transition reduces the potential. For ⟨combine⟩, we need to show $\Phi(2^i) + \Phi(2^i) > \Phi(2^{i+1})$, which, depending on i, reduces to either $2 + 2 > 2$ or $2^j + 1 + 2^j + 1 > 2^{j+1} + 1$ for some $j \geq 0$. In the case of ⟨modulo⟩, we note that $2m \geq 2^d$ (as $d = \lceil \log_2 m \rceil$), so $2^d - m \leq 2^0 + 2^1 + ... + 2^{d-2}$. It thus suffices to show $\Phi(2^d) \geq \sum_{i=0}^{d-2} \Phi(2^i) + (d - 2)$, and we get

$$\sum_{i=0}^{d-2} \Phi(2^i) = 2d' + (d - d' - 2) + 2^{d-d'-1} - 1 \leq 2d + 2^{d-d'-1}$$

So $\Phi(2^d) - \sum_{i=0}^{d-2} \Phi(2^i) \geq 2^{d-d'} + 1 - (2d + 2^{d-d'-1})$ which is at least $d - 2$ if $2^{d-d'-1} \geq 3d - 3$. The latter then follows from our choice of d'. Finally, ⟨fast modulo⟩ obviously reduces the potential as well. ◀

Apart from remainder predicates, the potential function is easy to choose. For threshold predicates, every state (except 0) has potential 1, and for an input state its potential is simply given by the total potential of the states it distributes to, plus one. The combined state 0 still has potential 0. This leads to the following statement.

▶ **Proposition 24.** *For every population computer $\mathcal{P}$ produced by Theorem 1, there exists a potential function Φ for $\mathcal{P}$. Additionally, $\Phi(0) = 0$ and $\Phi(q) \in \mathcal{O}(|\varphi|)$ for all $q \in Q$.*

Rapidly-Decreasing

Our next goal is showing that the potential function Φ constructed in the previous section is rapidly-decreasing.

Throughout this section we will need to reference particular states or sets of states. First, recall that I are the input states, 0 is the combined reservoir state, and we have s

subcomputers, each of which corresponds to either a remainder or a threshold predicate φ_j. Subcomputer j has a degree d_j, for $j = 1, ..., s$, corresponding to the bits of the representation it encodes. We need to reference specifically the highest bits of threshold subcomputers, so let $Q_d := \{(2^{d_j})_j : \varphi_j \text{ is threshold predicate}\}$.

Additionally, we are interested in the states q for which a transition using only agents in q exists, which are precisely $Q_{\text{self}} := Q \setminus (I \cup Q_d \cup \{0\})$.

We start by showing two technical lemmata. It will be necessary to keep track of which state contributes to which subcomputer, which we want to denote by $\text{value}_j(q)$, for $q \in Q$ and a subcomputer $j \in \{1, ..., s\}$. In particular, we set $\text{value}_j((q)_j) := q$ for each $q \in Q_j$, $\text{value}_j(0) := 0$ and $\text{value}_j(x_i) := a_i^j$ for input $i \in \{1, .., v\}$.

Note that the $\sum_q \text{value}_j(q)C(q)$ is invariant for configurations C of a run, if j is a threshold predicate. (For remainder predicates it would be invariant modulo m_j, but that is not relevant for this section.) Also, the sum $\sum_q |\text{value}_j(q)|C(q)$ is nonincreasing, for all j.

First, we prove that there cannot be too many agents in Q_d; they would hold too much value.

▶ **Lemma 25.** *Let C denote a reachable configuration. Then $C(Q_d) \leq |C|/8$.*

Proof. Let j denote the index of the threshold predicate, $q = (\pm 2^{d_j})_j \in Q_d$ one of its largest states, and $p \in I$ an input state. We then have

$$|\text{value}_j(p)| \leq a^j_{\max} \leq \frac{2^{d_j}}{16s} = \frac{\text{value}_j(q)}{16s}$$

due to the choice of d_j. As C is reachable from some initial configuration C_0, and $\sum_q |\text{value}_j(q)|C(q)$ cannot increase, we sum over p to get

$$|\text{value}_j(q)|\, C(q) \leq \sum_{p\in I} |\text{value}_j(p)|\, C(p) \leq \sum_{p \in I} |\text{value}_j(q)|\, C(p)/16s = |\text{value}_j(q)|\, C(I)/16s$$

So we have $C(q) \leq C(I)/16s$ and summing over q yields the desired statement. ◀

▶ **Lemma 26.** *Let $t_j := ((2^{d_j})_j, (-2^{d_j})_j \mapsto 0, 0) \in \langle \textsf{cancel} \rangle$, where j is the index of a threshold subcomputer. Let C, C' denote configurations with $C \to C'$.*

Then $C(Q_d) - C'(Q_d) \leq 2 \sum_j \text{tmin}_{t_j}(C) + C(I) + C(Q_{\text{self}})$.

Proof. Let $D \in \mathbb{N}^{Q_d}$ with $D(q) := \max\{C(q) - C'(q), 0\}$ for $q \in Q_d$. Note that $D(q)$ is a lower bound on how many agents leave state q in any run from C to C', and that $C(Q_d) - C'(Q_d) \leq D(Q_d)$.

Let j be the index of a threshold subcomputer. As the only way to leave these states are $\langle \textsf{cancel} \rangle$ and $\langle \textsf{cancel 2nd highest} \rangle$, we know that

$$\sum_{q \in Q_{j+}} |\text{value}_j(q)|\, C(q) \geq 2^{d_j - 1}\Big(D((2^{d_j})_j) + D((-2^{d_j})_j)\Big)$$

where $Q_{j+} := \{q \in Q : \text{value}_j(q) > 0\}$. The same inequality holds when replacing Q_{j+} with $Q_{j-} := \{q \in Q : \text{value}_j(q) < 0\}$. From these, we derive

$$2\, C((\sim 2^{d_j})_j) + 2 \sum_{q \in Q \setminus Q_d} 2^{-d_j} |\text{value}_j(q)|\, C(q) \geq D((2^{d_j})_j) + D((-2^{d_j})_j)$$

for $\sim \in \{+, -\}$, as $Q \setminus Q_d \supseteq Q_{j\sim} \cup \{(-2^{d_j})_j\}$. We can combine the two inequalities into

$$2\, \text{tmin}_{t_j}(C) + 2 \sum_{q \in Q \setminus Q_d} 2^{-d_j} |\text{value}_j(q)|\, C(q) \geq D((2^{d_j})_j) + D((-2^{d_j})_j)$$

Summing over j then yields

$$2\sum_{t\in T_d}\mathrm{tmin}_t(C)+2\sum_{q\in Q\setminus Q_d}\sum_j 2^{-d_j}|\mathrm{value}_j(q)|\,C(q)\geq D(Q_d)$$

If $\sum_j 2^{-d_j}|\mathrm{value}_j(q)|\leq\frac{1}{2}$ were to hold for all $q\in Q\setminus Q_d$, then we would get the desired statement (noting value$(0)=0$), so it remains to show this claim. For $q\in Q_j\setminus Q_d$ for some j we have $|\mathrm{value}_j(q)|\leq 2^{d_j-1}$ and $\mathrm{value}_{j'}(q)=0$ for $j'\neq j$, and for $q\in I$ it follows from $|\mathrm{value}_j(q)|\leq a^j_{\max}\leq 2^{d_j}/16s$ for all $j\in\{1,...,s\}$. ◀

▶ **Lemma 21.** Φ *is* $\mathcal{O}(|\varphi|^3)$*-rapidly decreasing in all well-initialised configurations.*

Proof. Let $C'\in\mathbb{Z}^Q$ with $C':=C-C_{\mathrm{term}}$ and let $W:=\max_{q\in Q}\Phi(q)$ denote the largest weight. Then

$$\Phi(C)-\Phi(C_{\mathrm{term}})=\Phi(C')\leq W\,C'(Q\setminus\{0\})$$

Applying Lemma 26, as well as $C'\leq C$, yields

$$C'(Q\setminus\{0\})=C'(Q_d)+C'(I)+C'(Q_{\mathrm{self}})\leq 2\sum_{t\in T_d}\mathrm{tmin}_t(C)+2C(I)+2C(Q_{\mathrm{self}}) \qquad (*)$$

where $T_d:=\{t_j:\varphi_j \text{ threshold predicate}\}$ with t_j as for Lemma 26. As we are in a well-initialised configuration, $C(I)\leq 2(C(Q_{\mathrm{self}})+C(0)+C(Q_d))$, and Lemma 25 implies $C(Q_d)\leq C(Q\setminus Q_d)/7$. We use both to derive

$$\begin{aligned}&C(I)\leq 2C(Q_{\mathrm{self}})+2C(0)+\tfrac{2}{7}\big(C(I)+C(Q_{\mathrm{self}})+C(0)\big)\\ \Rightarrow\ &\tfrac{5}{7}C(I)\leq\tfrac{16}{7}(C(Q_{\mathrm{self}})+C(0))\\ \Rightarrow\ &C(I)\leq\tfrac{16}{5}\big(C(Q_{\mathrm{self}})+C(0)\big)\leq 4\big(C(Q_{\mathrm{self}})+C(0)\big)\end{aligned}$$

Of course, $C(I)\leq 4C(Q_{\mathrm{self}})+4\min\{C(0),C(I)\}$ is thus true as well. We write $T_I:=\langle\mathsf{distribute}\rangle$ and use Lemma 63 to get

$$C(I)\leq 4C(Q_{\mathrm{self}})+4\sum_{t\in T_I}\mathrm{tmin}_t(C)$$

Every state $q\in Q_{\mathrm{self}}$ has a (unique) transition using only agents in q; we use T_o to denote this set and set $T:=T_I\cup T_o\cup T_d$. We can now insert the previous inequality into $(*)$ to get $C'(Q\setminus\{0\})\leq 8\sum_{t\in T}\mathrm{tmin}_t(C)$. Noting $|T|\leq 2|Q|$, we finally apply Lemma 66.

$$\Phi(C')^2\leq\Big(8\,W\sum_{t\in T}\mathrm{tmin}_t(C)\Big)^2\leq 128\,W^2|Q|\,\mathrm{speed}(C)$$

The desired statement then follows from $|Q|,W\in\mathcal{O}(|\varphi|)$. ◀

C Conversion 1: Removing Multiway Transitions

In this section we describe how to convert a population computer $\mathcal{P} = (Q, \delta, I, O, H)$ with arbitrary interactions into a population computer $\mathcal{P}' = (Q', \delta', I', O', H')$ where all interactions are binary. The goal is to show the following theorem. (We remark that the definition of potential function can be found in Section 7.4.)

▶ **Theorem 27.** *Let $\mathcal{P} = (Q, \delta, I, O, H)$ be a bounded population computer. Then there exists an equivalent bounded binary population computer $\mathcal{P}' = (Q', \delta', I', O', H')$ with adjusted size $\mathcal{O}(\beta \cdot \text{size}(P))$, where $\beta \leq |Q|$. Additionally:*

1. *If every state of $\mathcal{P}$ but one has at most 2 outgoing transitions, $\beta \leq 2$.*
2. *If no state in I has incoming transitions, then neither do states in I'.*
3. *If all configurations in $\mathbb{N}^I$ are terminal and $r(q) \leq 1$ for $q \in I$ and $(r \mapsto p) \in \delta$, then all configurations in $\mathbb{N}^{I'}$ are terminal.*
4. *If there is a potential function Φ for $\mathcal{P}$ which is α-rapidly decreasing in all well-initialised configurations, and conditions 2 and 3 are met, then there is a potential function Φ' for $\mathcal{P}'$, which is $\mathcal{O}(|Q|^2 k^2 \alpha)$-rapidly decreasing in all well-initialised configurations, where $k := \max\{|r| : (r \mapsto p) \in \delta\}$.*

We remark that rapid population computers fulfil all four conditions by definition.

The theorem contains all properties to make this section self-contained. Its proof is split into lemmata 30, 31, 32 for the basic properties, Lemma 33 for conditions 1-3, and lemmata 37 and 38 for condition 4.

This section is split into three parts. In Appendix C.1 we describe the construction formally, in Appendix C.2 we show the desired properties except for speed, which is proven in Appendix C.3.

C.1 Construction

Let $m(q) := \max\{r(q) : (r \mapsto s) \in \delta\}$ denote the maximum multiplicity of any outgoing transition of q. For each state q we allow up to $m(q)$ agents to "stack", so we create states $\{(q, i) : q \in Q, i = 0, ..., m(q)\}$ and the following transitions, for $q \in Q$, $i, j \in \{1, ..., m(q) - 1\}$.

$$\begin{aligned} &(q, i), (q, j) \mapsto (q, i + j), (q, 0) && \text{if } i + j \leq m(q) \\ &(q, i), (q, j) \mapsto (q, m(q)), (q, i + j - m(q)) && \text{if } i + j \geq m(q). \end{aligned} \qquad \langle \mathsf{stack} \rangle$$

Intuitively, an agents in (q, i) „owns“ i agents in state q, meaning that it certifies that $i - 1$ additional agents are in $(q, 0)$. A transition $t = (r \mapsto s) \in \delta$ with $\text{supp}(r) = \{q, p\}$ can be initiated by any pair of agents, who together represent at least r agents, i.e. an agent in (q, i) with $i \geq r(q)$ and an agent in (p, j) with $j \geq r(p)$ (if $p \neq q$, else we need $i + j \geq r(q)$).

One of these agents will be designated as the primary agent, responsible for executing the remainder of the transition. This is based on their corresponding state in Q; here we will assume q to be primary. First, the secondary agent in state (p, j) reduces its value by $r(p)$, moving to $(p, j - r(p))$, while the primary agent moves to $(q, i - r(q), t)$. This indicates that it has ownership of $i - r(q)$ agents in q, and is in the process of executing transition t. We then have states $\{(q, i, t) : i = 0, ..., m(q)\}$, and, assuming $q \neq p$, transitions

$$(q, i), (p, j) \mapsto (q, i - r(q), t), (p, j - r(p)) \qquad \text{for } i \geq r(q), j \geq r(p) \qquad \langle \mathsf{commit} \rangle$$

If $q = p$, we pick any i, j with $i + j \geq r(q)$ and get

$$\begin{aligned} &(q,i),(q,j) \mapsto (q, i+j-r(q), t), (q, 0) && \text{if } i+j-r(q) \leq m(q) \\ &(q,i),(q,j) \mapsto (q, i+j-r(q)-m(q), t), (q, m(q)) && \text{else} \end{aligned} \qquad \langle\textsf{commit}\rangle$$

Afterwards, the agent in $(q, i-r(q), t)$ needs to transfer its remaining value (if any) of $i-r(q)$ to another agent.

$$(q,i,t),(q,0) \mapsto (q,0,t),(q,i) \qquad \text{for } i = 1, ..., m(q) \qquad \langle\textsf{transfer}\rangle$$

Finally, the transition can be executed. For this, we add states $\{(t,i) : i = 1, ..., |r|\}$. Let $s_1, ..., s_l$ denote an enumeration of the resulting agents s, with $l := |s|$. Intuitively, an agent in (t,i) moves one agent into $(s_i, 1)$, and then goes to $(t, i+1)$. Instead of moving an agent into $(t,1)$ via a transition, we identify $(q,0,t)$ with $(t,1)$ directly. Additionally, we identify (t,l) with $(s_l, 1)$, so that we do not have to create a special transition for the last agent. The remaining transitions are as follows, for $i = 1, ..., l-1$.

$$\begin{aligned} &(t,i),(p,0) \mapsto (t,i+1),(s_i,1) && \text{if } i \leq r(p) \\ &(t,i),(q,0) \mapsto (t,i+1),(s_i,1) && \text{if } i > r(p) \end{aligned} \qquad \langle\textsf{execute}\rangle$$

As defined above, $\mathcal{P}'$ is not deterministic, i.e. for some of the transitions $\langle\textsf{stack}\rangle$ and $\langle\textsf{commit}\rangle$ it may be the case that $\langle\textsf{stack}\rangle = (r \mapsto s_1)$ and $\langle\textsf{commit}\rangle = (r \mapsto s_2)$ for the same r. However, in this case we always prefer to execute $\langle\textsf{commit}\rangle$ over $\langle\textsf{stack}\rangle$, and, in the case of conflicting $\langle\textsf{commit}\rangle$ transitions, pick an arbitrary one.

We retain the original input states and helpers, by identifying each $q \in Q$ with $(q, 1)$. For the output function we define $O'(S) := O(\{q : (q,i) \in S\})$ for any S, but note that a circuit for O grows by at most a factor of 3, as $(q,i) \in \text{supp}(C) \Rightarrow (q,0) \in \text{supp}(C)$ for $i \geq 2$ and any reachable configuration C and state q, so it suffices to check for $(q,0)$ and $(q,1)$.

C.2 Correctness

We want to show that $\mathcal{P}'$ is bounded and decides the same predicate as $\mathcal{P}$. To this end, we will show that $\mathcal{P}'$ "refines" $\mathcal{P}$, i.e. that runs of $\mathcal{P}'$ correspond to runs of $\mathcal{P}$ in some fashion. The formal definition of refinement is given in Appendix I.1.

To begin, let us introduce the mapping between configurations of $\mathcal{P}'$ and $\mathcal{P}$ describing the refinement. We define $\pi : Q' \to \mathbb{N}^Q$ by setting

$$\begin{aligned} \pi((q,i)) &:= q \cdot i \\ \pi((q,i,t)) &:= q \cdot i + s && \text{for all } q \in Q,\ t = (r \mapsto s) \in \delta \text{ and } i \\ \pi((t,i)) &:= s_i + ... + s_l \end{aligned}$$

This uses the same enumeration of s as above. Clearly, π is well-defined, as $\pi((q,0,t)) = s = \pi((t,1))$ and $\pi((t,l)) = s_l = \pi((s_l,1))$. Finally, we extend π to a linear mapping $\pi : \mathbb{N}^{Q'} \to \mathbb{N}^Q$ in the obvious fashion.

Before we show that $\mathcal{P}'$ is a refinement, we argue that agents cannot get stuck in intermediate states, implying that $\mathcal{P}'$ can always make a transition at C if one is enabled at $\pi(C)$.

▶ **Lemma 28.** *Let $C \in \mathbb{N}^{Q'}$ be both reachable and terminal. Then $\pi(C)$ is terminal.*

Proof. We first observe that transition ⟨transfer⟩ is always enabled if an agent is in state (q, i, t), with $t = (r, s) \in \delta$ and $i \geq 1$, as that state "owns" $i + r(q)$ agents in q. Hence there must be $i + r(q) - 1 \geq i \geq 1$ agents in $(q, 0)$. By the same line of argument, ⟨execute⟩ is always enabled if an agent is in (t, i), for $t \in \delta$ and $1 \leq i < |s|$.

Now, assume $\pi(C)$ is not terminal, so there is some transition $t = (r \mapsto s) \in \delta$ which is enabled at $\pi(C)$. As we have just argued, all agents of C are in states (q, i), for $q \in Q$ and $i \in \{0, ..., m(q)\}$. Transition ⟨commit⟩ is not enabled at C, wherefore one of the states $q \in Q$ used by t fulfils $i < r(q)$ for all i with $C((q, i)) > 0$. But t is enabled at $\pi(C)$, so $\pi(C)(q) \geq r(q)$ and, by definition of π, there are $i, j > 0$ s.t. C contains both an agent in (q, i) and one in (q, j). Due to our choice of m, we have $r(q) \leq m(q)$, and therefore $0 < i, j < m(q)$, wherefore transition ⟨stack⟩ is enabled, contradicting that C is terminal. ◀

▶ **Lemma 29.** *$\mathcal{P}'$ refines $\mathcal{P}$.*

Proof. We show that π fulfils the properties required by Definition 61.

1. Note that π is invariant under execution of ⟨stack⟩, ⟨transfer⟩ and ⟨execute⟩. Additionally, for a $t \in \delta$ and a corresponding ⟨commit⟩ transition t', we find that $C \rightarrow_{t'} C'$ implies $\pi(C) \rightarrow_t \pi(C')$ for all $C, C' \in \mathbb{N}^{Q'}$.
2. We identified $q \in Q$ with $(q, 1)$ and set $\pi((q, 1)) := q$, so $I = I'$ and $\pi(C) = C$ for all $C \in \mathbb{N}^Q$ follows.
3. This follows immediately from Lemma 28 and our choice of O'.

◀

To conclude the proof, we merely need to show that $\mathcal{P}'$ is bounded and can then rely on the properties of refinement.

▶ **Lemma 30.** *$\mathcal{P}'$ is bounded.*

Proof. Assume an infinite run $C_0, C_1, ...$ of $\mathcal{P}'$ exists. If transition ⟨commit⟩ is executed infinitely often in that run, at steps $i_0, i_1, ... \in \mathbb{N}$, then $\pi(C_{i_0}), \pi(C_{i_1}), ...$ would be an infinite run of $\mathcal{P}$, contradicting that $\mathcal{P}$ is bounded. Hence there is an infinite suffix of $C_0, C_1, ...$ in which ⟨commit⟩ is never fired.

In this suffix the number of agents in a state $(q, i, t) \in Q'$ with $i > 0$ cannot increase, but decreases whenever ⟨transfer⟩ is executed. Hence this also happens only finitely often and the number of agents in a state (t, i) cannot increase beyond a point. As ⟨execute⟩ increases i, it too must occur only finitely often. The only transition left is ⟨stack⟩, which always increases the number of agents in either $(q, 0)$ or $(q, m(q))$, for some $q \in Q$. ◀

▶ **Lemma 31.** *If $\mathcal{P}$ decides φ, then $\mathcal{P}'$ does so as well.*

Proof. This follows immediately from lemmata 29, 30 and 62. ◀

To estimate the size of the resulting protocol, let β_q denote the number of transitions $t \in \delta$ for which $q \in Q$ is the primary agent, and set $\beta := \max\{\beta_q : q \in Q\}$.

▶ **Lemma 32.** $|Q'| \leq (\beta + 2)\,\text{size}(\mathcal{P})$.

Proof. We begin by bounding the total value of $m(q)$. Clearly, $m(q) \leq \sum_{(r \mapsto s) \in \delta} r(q)$ for $q \in Q$, and thus $\sum_{q \in Q} m(q) \leq \sum_{t \in \delta} |t|$.

For every $q \in Q$ we create $m(q) + 1$ states, for every $t \in \delta$ we create $|t|$ states, and we create an additional $m(q)$ states for every transition t using state q as primary agent, so at most $\beta m(q)$. In total we create at most $(\beta + 2) \sum_{t \in \delta} |t| + |Q|$ states. ◀

▶ **Lemma 33.**
- *If every state of $\mathcal{P}$ but one has at most 2 outgoing transitions, $\beta \leq 2$.*
- *If no state in I has incoming transitions, then neither do states in I'.*
- *If all configurations in $\mathbb{N}^I$ are terminal and $r(q) \leq 1$ for $q \in I$ and $(r \mapsto p) \in \delta$, then all configurations in $\mathbb{N}^{I'}$ are terminal.*

Proof. To show the first statement, let $q \in Q$ denote the state with the most outgoing transitions. For our construction, we can simply not choose q as primary agent, for all transitions that also uses a different agent. There is at most one other transitions (using only agents in q), so every state is chosen as primary agent with at most 2 transitions and $\beta \leq 2$.

The second statement is obvious from the construction.

For the third, note that the condition implies $m(q) \leq 1$ for all $q \in I$. Therefore the only transition using an agent in I' is $\langle \mathsf{commit} \rangle$, which is not enabled at C if $\pi(C)$ is terminal. ◀

C.3 Speed

In this section we want to show that our construction is fast, by proving that a potential function exists and that it is rapidly decreasing. We assume that Condition 4 of Theorem 27 is met (and thus conditions 2 and 3 as well).

The next lemma constructs a potential function for $\mathcal{P}'$, based on the potential of $\mathcal{P}$. The potential of a state then corresponds directly to the original potential of states it owns, with some additional accounting to pay for overhead of executing a multiway transition.

At this point it becomes important that our definition of potential function requires a transition of size k to reduce the potential by $k-1$, as this means that we have to increase the total potential by only a constant factor.

▶ **Lemma 34.** *Let Φ be a potential function for $\mathcal{P}$. There is a potential function Φ' for $\mathcal{P}'$.*

Proof. We first adjust Φ by multiplying it with 5, so that $\Phi(r) \geq \Phi(s) + 5(|s|-1)$ for all transitions $(r \mapsto s) \in \delta$.

Let $q, p \in Q$, $t = (r \mapsto s) \in \delta$ a transition where $\operatorname{supp}(r) = \{q, p\}$ and q is primary, and let $s_1, ..., s_l$ denote the enumeration of s from above. We define Φ' as

$$\begin{aligned}
\Phi'((q,0)) &:= 0 \\
\Phi'((q,i)) &:= i\Phi(q) + 1 && \text{for } i \in \{1, ..., m(q)-1\} \\
\Phi'((q,m(q))) &:= m(q)\Phi(q) \\
\Phi'((q,i,t)) &:= i\Phi(q) + \Phi(s) + 2l + 2 && \text{for } i \in \{1, ..., m(q)\} \\
\Phi'((t,i)) &:= \sum_{j=i}^{l} (\Phi(s_j) + 2) && \text{for } i \in \{1, ..., l-1\}
\end{aligned}$$

For most transitions, it is easy to see that Φ' decreases. However, we need to verify that $\langle \mathsf{commit} \rangle$ does so as well. If $q \neq p$, we have to prove the inequality

$$i\Phi(q) + j\Phi(p) + 2 \geq (i - r(q))\Phi(q) + \Phi(s) + 2l + 2 + (j - r(p))\Phi(p) + 1$$

which boils down to $\Phi(r) \geq \Phi(s) + 2l + 1$. This then follows from $\Phi(r) \geq \Phi(s) + 5(l-1)$. The case $q = p$ is shown analogously. ◀

The next three lemmata show technical properties that are needed for the proof that Φ' is rapidly decreasing. The first gives a relation between the potential of $\mathcal{P}'$ and of the refined computer $\mathcal{P}$.

▶ **Lemma 35.** $\Phi(\pi(C)) \leq \Phi'(C) \leq \Phi(\pi(C)) + 2C(S)$ *for* $S := Q' \setminus \{(q, m(q)), (q, 0) : q \in Q\}$.

Proof. For the first inequality we simply observe $\Phi(\pi(C)) \leq \Phi'(q)$ for each $q \in Q'$.

Each state $q = (p, m(p))$, for $p \in Q$, has $\Phi'(q) = \Phi(\pi(q))$. For each other state $q \in S$ we show $\Phi'(q) - \Phi(\pi(q)) \leq 2c$, where c is the amount of agents "owned" by an agent in state q. E.g. (q, i) for $q \in Q$ and $0 < i < m(q)$ has $\Phi'(q) - \Phi(\pi(C)) = 1$ and $c = i$. For (q, i, t), we have $c = |s| + i$, and (t, i) owns $c = |s| + 1 - i$ agents, the respective inequalities follow immediately. ◀

In prior proofs we have seen that states which can initiate a transitions by themselves are useful to show speed bounds. More concretely, this means that there is a state q and a transition $t = (r \mapsto s)$ with $\mathrm{supp}(r) = \{q\}$, implying $\mathrm{tmin}_t(C) = C(q)$ for all C. For the next lemma, we show that there are transitions with the same useful property (even if not all of them use only a single state).

▶ **Lemma 36.** *Let* $S := Q' \setminus \{(q, m(q)), (q, 0) : q \in Q\}$. *There is an injection* $g : S \to \delta'$, *s.t.* $C(q') = \mathrm{tmin}_{g(q')}(C)$ *for any reachable configuration* C *and* $q' \in S$.

Proof. Let $q \in Q$, and $t = (r \mapsto s) \in \delta$.

If $q' = (q, i)$ for $i < m(q)$, then there is a ⟨stack⟩ transition using only agents in q, which we use as $g(q')$. If $q' = (q, i, t)$, then we know that $C(q') \leq C((q, 0))$, as q' owns at least $|r| \geq 1$ agents, so we can pick the ⟨transfer⟩ transitions for $g(q')$. Finally, if $q' = (t, i)$, for $i < |s|$, then q' owns $|s| - i \geq 1$ agents other than itself, and we choose the corresponding ⟨execute⟩ transition. ◀

In the end, we want to show that Φ' is rapidly decreasing in all well-initialised configurations if Φ is. For this, we need to argue briefly that being a well-initialised configuration corresponds.

▶ **Lemma 37.** *If* C *is well-initialised, then so is* $\pi(C)$.

Proof. Due to Condition 3 of Theorem 27, $m(q) = 1$ for each $q \in I$. In combination with Condition 2 we get $C((q, 1)) = \pi(C)(q)$ for all $q \in I$. This implies $C(I') = \pi(C)(I)$. Noting $|H| = |H'|$, the statement follows immediately. ◀

Finally, we can prove that the potential is rapidly decreasing.

▶ **Lemma 38.** *Let* $k := \max\{|r| : (r \mapsto s) \in \delta\}$. Φ' *is* $\mathcal{O}(\alpha k^2 |Q|^2)$*-rapidly decreasing in* C *if* Φ *is* α*-rapidly decreasing in* $\pi(C)$, *for all reachable configurations* C *and* $\alpha \geq 1$.

Proof. Let C_{term} denote a terminal configuration reachable from C. Using Lemma 35 we get $\Phi'(C) - \Phi'(C_{\mathrm{term}}) \leq \Phi(\pi(C)) - \Phi(\pi(C_{\mathrm{term}})) + 2C(S)$. We know that $\pi(C_{\mathrm{term}})$ is reachable from $\pi(C)$, and, as shown in the proof of Lemma 31, it is terminal as well. Hence we can use that $\mathcal{P}$ is α-rapidly decreasing in $\pi(C)$ to get $\big(\Phi(\pi(C)) - \Phi(\pi(C_{\mathrm{term}}))\big)^2 \leq \alpha\,\mathrm{speed}(\pi(C))$.

To estimate $\mathrm{speed}(\pi(C))$, we write t^* for the ⟨commit⟩ transition corresponding to $t \in \delta$ using agents in states $(q, m(q)), (p, m(p))$ for $q, p \in Q$. Additionally, let $H_q := \sum_{q' \in S} C(q')\pi(q')(q)$ for $q \in Q$ denote the contribution of agents in states S to $\pi(C)(q)$, for $q \in Q$. We then observe $\pi(C)(q) = H_q + m(q)C((q, m(q))) \leq H_q + kC((q, m(q)))$.

Now, let $t = (r \mapsto s) \in \delta$ and $q, p \in Q$ with $\mathrm{supp}(r) = \{q, p\}$. Then the above (together with Lemma 65) yields

$$\mathrm{tmin}_t(\pi(C)) = \min\{\pi(C)(q), \pi(C)(p)\} \leq k\,\mathrm{tmin}_{t^*}(C) + H_q + H_p.$$

Via Lemma 66, squaring the right-hand side gives at most $3(k^2 \operatorname{tmin}_{t^*}(C)^2 + H_q^2 + H_p^2)$, so by summing over t we get

$$\operatorname{speed}(\pi(C)) \le 3k^2 \sum_{t\in\delta} \operatorname{tmin}_{t^*}(C)^2 + 6|Q| \sum_{q\in Q} H_q^2$$

Moving on, we get $\sum_{q\in Q} H_q^2 \le \left(\sum_{q\in Q} H_q\right)^2$ and $\sum_{q\in Q} H_q = |\pi(C_S)|$, where $C_S(q) := C(q)$ for $q \in S$ and 0 otherwise. Based on the definition of π, we have $|\pi(q)| \le 2k$, so $|\pi(C_S)| \le 2k|C_S|$. Due to Lemma 36 we find $|C_S| \le \sum_{t'\in g(S)} \operatorname{tmin}_{t'}(C)$.

Putting things together, and applying Lemma 66 again, we get the following bound.

$$\operatorname{speed}(\pi(C)) \le 3k^2 \sum_{t\in\delta} \operatorname{tmin}_{t^*}(C)^2 + 24k^2|Q|^2 \sum_{t'\in g(S)} \operatorname{tmin}_{t'}(C)^2 \le 24k^2|Q|^2 \operatorname{speed}(C)$$

Now we can go back to the start, and complete the proof. (Note $C(S) = |C_S|$.)

$$\begin{aligned}&\big(\Phi'(C) - \Phi'(C_{\text{term}})\big)^2 \le 2\big(\Phi(\pi(C)) - \Phi(\pi(C_{\text{term}}))\big)^2 + 8C(S)^2\\ &\le 2\alpha \operatorname{speed}(\pi(C)) + 8|Q| \operatorname{speed}(C) \le 49\alpha k^2|Q|^2 \operatorname{speed}(C)\end{aligned}$$

◀

D Conversion 2: Converting Output functions to Marked Consensus

In this section, we show how to convert a bounded binary computer $\mathcal{P}$ into an equivalent computer $\mathcal{P}'$ using a marked consensus output instead of a general output function. We prove the following conversion theorem, consisting of multiple independent parts. The last part concerns the speed analysis, the relevant definitions can be found in Appendix H.

▶ **Theorem 39.** *Let $\mathcal{P} = (Q, \delta, I, O, H)$ be a bounded binary population computer. Then there exists an equivalent bounded binary computer $\mathcal{P}' = (Q', \delta', I', O', H')$ with adjusted size $\mathcal{O}(\text{size}_2(\mathcal{P}))$ using a marked consensus output. Additionally:*

1. *If no state in I has incoming transitions, then neither do states in I'.*
2. *If all configurations in $\mathbb{N}^I$ are terminal, then so are all configurations in $\mathbb{N}^{I'}$.*
3. *If there is a potential function Φ for $\mathcal{P}$ which is α-rapidly decreasing in all well-initialised configurations, and Condition 2 is met, then $\mathcal{P}'$ has a potential group of size 5 which is $\mathcal{O}(\alpha + \text{size}_2(\mathcal{P})^2)$-rapidly decreasing in all well-initialised configurations.*

Theorem 39 summarises all the information necessary about the computers such that the rest of the conversions proceed smoothly. Specifics about the computer are not necessary for the next sections.

The proof will span this section and is split over lemmata 42 and 43 for the base statement, Lemma 44 for conditions 1 and 2, and Lemma 45 for Condition 3.

D.1 Construction

We begin by describing the construction that is depicted in Figure 4. Let $\mathcal{P} = (Q, \delta, I, O, H)$ denote a bounded binary population computer, and let (G, E) denote a circuit for O, where $G = \{1, ..., \beta\}$ is a set of NAND-gates, and $E : G \to (G \cup Q)^2$ specifies the two inputs for each gate. We assume that $u < g$ for all $g \in G$ and $u \in E(g) \cap G$, and that gate β gives the output of the circuit.

The computer $\mathcal{P}'$ consists of four parts:

- $Q_{\text{orig}} := Q \times \{-, +\}$, the states of the original protocol combined with a flag to indicate that the support might have changed.
- $Q_{\text{supp}} := Q \times \{0, 1, !\}$, one agent for each state of $\mathcal{P}$, detecting whether that state is present. ! is equivalent to 1, but also indicates that the gates must be reset.
- $Q_{\text{gate}} := G \times \{0, 1, \perp\}^3$, one agent for each gate, which stores the current values of its two inputs and its output. Uninitialised values are stored as $\perp$.
- $Q_{\text{reset}} := \{0, ..., |Q| + \beta\}$, one agent that resets the agents in Q_{supp} if the support changed, and which resets the agents in Q_{gate}, if the output of the circuit needs to be recomputed.

We now formally specify the required transitions. First, we need to refine the original protocol, requesting to recompute the support with each transition. Different occurrences of $\pm$ need not match.

$$(q, \pm), (p, \pm) \mapsto (q', +), (p', -) \qquad \text{for } (q, p \mapsto q', p') \in T \qquad \langle\text{execute}\rangle$$

It suffices to reset once, so for the purpose of speed we clear superfluous flags.

$$(q, +), (p, +) \mapsto (q, +), (p, -) \qquad \text{for } q, p \in Q \qquad \langle\text{denotify}\rangle$$

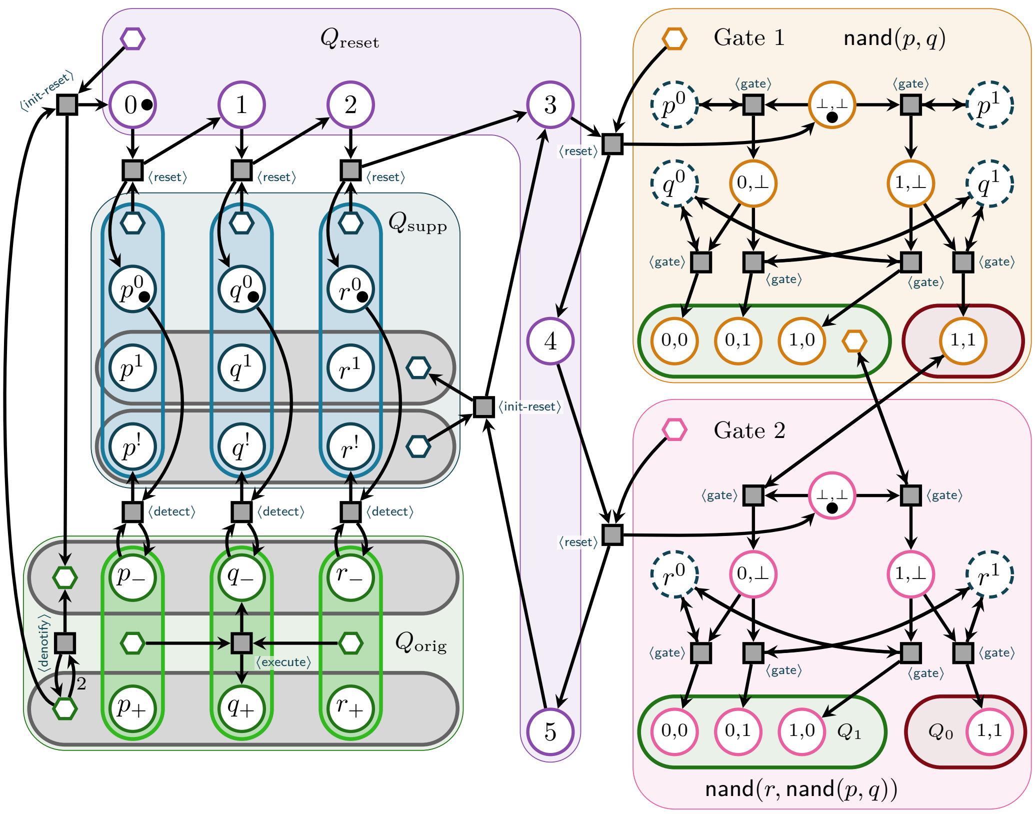

Figure 4 Figure visualising the conversion to marked consensus output for a population computer with states $\{p, q, r\}$, a single transition $p, q \mapsto r, r$ and output function $\neg(r \wedge \neg(p \wedge q))$.
State names are abbreviated: x^y and x_y are used instead of (x, y), and only the last two components of states in Q_{gate} are shown. Transitions are drawn using Petri net notation. The same state may appear multiple times. Occurrences beyond the first are drawn with a dashed border.
Hexagons indicate wildcards. Each hexagon is associated with a group of states (indicated by shaded areas). Used as input to a transition, it denotes that any agent of that group can be used (i.e. they create multiple copies of the transition). Transition may use hexagons as both input and output, in which case the output hexagon refers to the state corresponding to the state used for the input hexagon. For example, transition ⟨denotify⟩ takes two agents x_+, y_+, with $x, y \in \{p, q, r\}$ and produces an agent in x_- and one in y_+.
We omit ⟨leader⟩ transitions and the third case of ⟨init-reset⟩ from the drawing. Please note, however, that the resulting computer is still correct under the assumption that no superfluous helpers are provided. (This is equivalent to using leaders instead of helpers.)

To keep the protocol deterministic, we remove all ⟨denotify⟩ transitions which could also initiate another transition (in particular ⟨execute⟩). The support is computed by setting the stored bit to '!' once the corresponding state has been observed.

$$(q, 0), (q, -) \mapsto (q, !), (q, -) \qquad \text{for } q \in Q \qquad \langle\text{detect}\rangle$$

To define the transitions for Q_{gate}, we need to introduce some notation. First, we write nand for the NAND function, i.e. $\text{nand}(i, j) := \neg(i \wedge j)$ for $i, j \in \{0, 1\}$ and $\text{nand}(i, j) := \bot$ otherwise. We also use E_1, E_2 to denote the first and second component of E, and write S_g^b for the set of states indicating that gate $g \in Q \cup G$ has truth value $b \in \{0, 1\}$, i.e. $S_q^b := \{(q, b)\}$ for $q \in Q$ and $S_g^b := \{g, b\} \times \{0, 1\}^2$ for $g \in G$. We add the following transitions, for any gate

$g \in G$ and $b, i \in \{0, 1\}$.

$$\begin{aligned} (g, \perp, \perp, \perp), q &\mapsto (g, \perp, b, i), q && \text{for } q \in S^b_{E_1(g)} \\ (g, \perp, i, \perp), q &\mapsto (g, \mathsf{nand}(i, b), i, b), q && \text{for } q \in S^b_{E_2(g)} \end{aligned} \qquad \langle\mathsf{gate}\rangle$$

These transitions perform the computation of the gate, by initialising the first and then the second input. Once the second input is initialised, the output of the gate is set accordingly.

There are two kinds of resets; depending on whether the agents in Q_{supp} are affected. (The gates are always reset.) Both resets are executed by an agent in Q_{reset}, who goes through the other agents one by one. Let $q_1, ..., q_{|Q|}$ denote an enumeration of Q.

$$\begin{aligned} i - 1, (q_i, b) &\mapsto i, (q_i, 0) && \text{for } (q_i, b) \in Q_{\text{supp}} \\ |Q| + g - 1, (g, b_1, b_2, b_3) &\mapsto |Q| + g, (g, \perp, \perp, \perp) && \text{for } (g, b_1, b_2, b_3) \in Q_{\text{gate}} \end{aligned} \qquad \langle\mathsf{reset}\rangle$$

There are three ways to initiate a reset:

$$\begin{aligned} (q, +), i &\mapsto (q, -), 0 && \text{for } (q, +) \in Q_{\text{orig}}, i \in Q_{\text{reset}} \\ (q, !), i &\mapsto (q, 1), \min\{i, |Q|\} && \text{for } (q, !) \in Q_{\text{supp}}, i \in Q_{\text{reset}} \\ i, j &\mapsto 0, (q_h, -) && \text{for } i, j \in Q_{\text{reset}} \end{aligned} \qquad \langle\mathsf{init\text{-}reset}\rangle$$

First, an agent in Q_{orig} may indicate that the support has changed, and everything will be reset. Second, an agent in Q_{supp} will request that all gates be reset whenever it changes its output. Third, if two agents are in Q_{reset}, the computation so far must be discarded and we continue with only one of them. The other moves into an arbitrary state $q_h \in Q$ with $H(q_h) > 0$, so it is given back to $\mathcal{P}$ to use for its computations. Picking a state with $H(q_h) > 0$, i.e. a helper state, ensures that this does not affect the correctness of $\mathcal{P}$.

Finally, all states in Q_{supp} and Q_{gate} also participate in a leader election, to ensure that there is only one agent in Q_{supp} for each $q \in Q$, and only one agent for each gate.

$$\begin{aligned} q, q' &\mapsto q, 0 && \text{for } q, q' \in Q_{\text{supp}} \text{ s.t. } q_1 = q'_1 \\ g, g' &\mapsto g, 0 && \text{for } g, g' \in Q_{\text{gate}} \text{ s.t. } g_1 = g'_1 \end{aligned} \qquad \langle\mathsf{leader}\rangle$$

These transitions indirectly cause a reset, by producing an agent in Q_{reset}.

It remains to define the inputs, helpers and outputs. For this, we identify a state $q \in Q$ with $(q, -) \in Q_{\text{orig}}$. We define $I' := I$, as well as $H'(q) := H(q)$ for $q \in Q$ and $H(q) := 1$ for $q \in Q \times \{0\} \cup G \times \{(\perp, \perp, \perp)\} \cup \{0\}$. To define the marked consensus output, we pick the special states $Q_0 = \{(\beta, 1)\} \times \{0, 1\}^2$ and $Q_1 := \{(\beta, 0)\} \times \{0, 1\}^2$.

D.2 Correctness

As for the previous construction in Appendix C.2 we will show that $\mathcal{P}'$ refines $\mathcal{P}$, formally defined in Appendix I.1. Hence we introduce a mapping $\pi : \mathbb{N}^{Q'} \to \mathbb{N}^Q$ to describe the configuration that $\mathcal{P}'$ is representing. For all C we define

$$\pi(C) := \sum_{q \in Q} q \cdot C((q, \pm)) + q_h \cdot (C(Q_{\text{supp}} \cup Q_{\text{gate}} \cup Q_{\text{reset}}) - |Q| - \beta - 1)$$

Recall that $q_h \in Q$ is the state to which superfluous agents are moved, as defined in $\langle\mathsf{init\text{-}reset}\rangle$. Eventually, we have exactly one agent for each state in Q, to detect the support, exactly one agent for each gate, and exactly one reset agent, so $|Q| + \beta + 1$ in total. Everything beyond that is superfluous and will be returned to q_h at some point.

▶ **Lemma 40.** *Let C denote a reachable configuration of $\mathcal{P}'$. Then $C(Q_{\text{reset}}) \geq 1$, $C(\{q\} \times \{0, 1, !\}) \geq 1$ for $q \in Q$, and $C(\{g\} \times \{0, 1, \bot\}^3) \geq 1$ for $g \in G$. If C is terminal, the above hold with equality.*

Proof. Let $S_q := \{q\} \times \{0, 1, !\}$ for $q \in Q$ and $G_g := \{g\} \times \{0, 1, \bot\}^3$ for $g \in G$. First, note that the sets of states Q_{reset}, S_q and G_g each contain at least one agent in an initial configuration (due to the choice of H'). Additionally, they cannot be emptied, as every transition removing agents from one of these sets also puts at least one agent back. (Using Petri net terminology, they are traps.)

If two agents are in Q_{reset}, the third part of ⟨init-reset⟩ is active and C is not terminal. Similarly, if two agents are in S_q, for some q, or two agents are in G_g, for some g, then ⟨leader⟩ can be executed. ◀

▶ **Lemma 41.** *$\mathcal{P}'$ refines $\mathcal{P}$.*

Proof. We show that π fulfils the properties required by Definition 61. The first two are simple.

1. Observe that $\pi(C)$ is changed only via transition ⟨execute⟩, and that this happens according to a transition $t \in T$.
2. $I = I'$ holds by construction and the remainder follows from $H(q_h) > 0$ and the definition of π.

Property 3 will take up the remainder of this proof. Let C denote a reachable, terminal configuration of $\mathcal{P}'$. Using Lemma 40 we get $\pi(C) = \sum_{q \in Q} q \cdot C((q, \pm))$. Therefore, if a transition $t \in \delta$ is enabled at $\pi(C)$, the corresponding ⟨execute⟩ transition is enabled at C. As C is terminal, so is $\pi(C)$.

Finally we have to argue $O'(\text{supp}(C)) = O(\text{supp}(\pi(C)))$. Due to Lemma 40 we know that in C we have exactly one agent in either $(q, 0)$, $(q, 1)$, or $(q, !)$. It cannot be in $(q, !)$, as then transition ⟨init-reset⟩ would be enabled.

If it were in $(q, 0)$ but $\pi(C)(q) > 0$, then transition ⟨detect⟩ would be enabled, so that cannot be the case either. Conversely, if it were in $(q, 1)$ but $\pi(C)(q) = 0$, then we also arrive at a contradiction: after the last agent left $(q, \pm)$ via ⟨execute⟩, it must have triggered a reset, which moved the agent to $(q, 0)$ (or, if there were multiple agents in $q \times \{0, 1, !\}$, ⟨leader⟩ would have triggered another reset later). But after that reset $\pi(C)(q) = 0$, so it is impossible to leave $(q, 0)$.

Therefore we find that the agents in Q_{supp} precisely indicate the support of $\pi(C)$. Whenever an agent in Q_{supp} changes its opinion (either due to a ⟨reset⟩ or ⟨detect⟩), all gates will be reset. So there is some point at which the opinions of agents in Q_{supp} have stabilised (in particular, all inequalities of Lemma 40 are tight, else there would be another reset) and the unique agent in Q_{reset} is in state $|Q|$, i.e. it is in the process of resetting all gates. As the gates are reset in order of some topological sorting (so a gate is reset after its inputs are), a gate will only assume a value after its inputs have stabilised and therefore compute the correct value according to the circuit. As the circuit computes $O(\text{supp}(\pi(C)))$, the statement follows. ◀

▶ **Lemma 42.** *$\mathcal{P}'$ is bounded.*

Proof. Due to Lemma 41 we know that $\pi(C)$ can change only finitely often, as $\mathcal{P}$ is bounded, and thus transition ⟨execute⟩ can be executed only finitely often. After that, the number of agents in a state $Q \times \{+\} \subseteq Q_{\text{orig}}$ cannot increase, but can always decrease using the first ⟨init-reset⟩ transition. So eventually no agents remain in those states.

Parallel to that, both $C(Q_{\text{supp}})$ and $C(Q_{\text{gate}})$ cannot increase. Whenever $C(Q_{\text{supp}}) > |Q|$ or $C(Q_{\text{gate}}) > \beta$, ⟨leader⟩ is enabled and decreases one of them, until $C(Q_{\text{supp}}) = |Q|$ and $C(Q_{\text{gate}}) = \beta$. Afterwards, ⟨leader⟩ cannot fire again (note Lemma 40), and $C(Q_{\text{reset}})$ cannot increase. If $C(Q_{\text{reset}}) > 1$, the third case of ⟨init-reset⟩ will reduce this number, until $C(Q_{\text{reset}}) = 1$.

To summarise, eventually no agents remain in $Q \times \{+\}$ and all inequalities of Lemma 40 become tight. At that point, the first and third case of ⟨init-reset⟩ are disabled, and it is not possible for the agent in Q_{reset} to lower its value to below β. Via ⟨reset⟩, it will thus eventually arrive at β, and the first part of ⟨reset⟩ cannot be executed again.

Once that happens, agents cannot enter states $Q \times \{0\} \subseteq Q_{\text{supp}}$, such that ⟨detect⟩ can never occur any more. This then causes ⟨init-reset⟩ to eventually be fully disabled, and then ⟨reset⟩ as well. Finally, transition ⟨gate⟩ can then fire only finitely often, and the protocol terminates. ◀

▶ **Lemma 43.** *If $\mathcal{P}$ decides φ, then $\mathcal{P}'$ does so as well.*

Proof. This follows immediately from lemmata 41, 42 and 62. ◀

To close out this section, we argue that the technical conditions are met.

▶ **Lemma 44.** *If no state in I has incoming transitions, then neither do states in I'. If all configurations in $\mathbb{N}^I$ are terminal, then so are all configurations in $\mathbb{N}^{I'}$.*

Proof. Note that $I' = I \times \{-\}$ by definition. If no state in I has incoming transitions, then it is not possible to put an agent into $I \times \{+\}$, as that happens only via ⟨execute⟩. Therefore transitions ⟨denotify⟩ or the first part of ⟨init-reset⟩ cannot move an agent from $(q, +)$ to $(q, -)$, for $q \in I$. Finally, note that $q_h \notin I'$, as $\text{supp}(H) \cap I = \emptyset$ by the definition of population computers. (So, to be precise, the statement only holds once we modify our construction to delete unused states and transitions.)

For the second part, note that the only transition which can execute from a configuration in $\mathbb{N}^{I'}$ is ⟨execute⟩, but that requires a transition in $\mathcal{P}$ which can execute in a configuration in $\mathbb{N}^I$. ◀

D.3 Speed

As mentioned at the start of this section, the definitions for speed analysis can be found in Appendix H. We will use the more general notion of potential groups rather than potential functions.

While it is possible to provide a potential function for $\mathcal{P}'$ based on a potential function for $\mathcal{P}$, this will result in large constants for the speed of the protocol. The reason for this lies in the nature of our computation, which proceeds in multiple phases.

As an example, take transition ⟨execute⟩. One of the resulting agents has its flag set to $+$, which may initiate a reset of every agent in $Q_{\text{supp}} \cup Q_{\text{gate}}$. To pay for this work, every transition of $\mathcal{P}$ would have to reduce the potential by $|Q| + \beta$. However, most of this cost would be wasted; only the last reset needs to be executed fully, and the other resets are likely to be interrupted before completion.

▶ **Lemma 45.** *Let Φ denote a potential function for $\mathcal{P}$ which is α-rapidly decreasing in all well-initialised configurations, and assume that states in I have no incoming transitions in $\mathcal{P}'$. Then there exists a potential group Φ' of size 5 for $\mathcal{P}'$, which is $\mathcal{O}(\alpha + |Q|^2 + \beta^2)$-rapidly decreasing in all well-initialised configurations.*

Proof. We will construct the potential group $\Phi' = (\Phi'_1, ..., \Phi'_5)$ and show that it is rapidly decreasing in all well-initialised configurations. So let C denote such a configuration, and let C_{term} denote a terminal configuration reachable from C.

For the sake of readability we will defer the definition of the Φ'_i until it is used. However, note that the definition will be independent of C.

The proof is going to proceed via case distinction based on the properties of C. For the i-th case we are going to show that Φ'_i is active. We are going to implicitly assume that prior cases are excluded, so the proof for case i is going to rely on the conditions for cases $1, ..., i-1$ not being met.

Case 1. $C(Q_{\text{supp}} \cup Q_{\text{gate}} \cup Q_{\text{reset}}) > |Q| + \beta + 1$. For Φ'_1, the goal is to show that the "leader elections" for each state happen quickly. We set $\Phi'_1(q) := 2$ for $q \in Q_{\text{supp}} \cup Q_{\text{gate}}$, $\Phi'_1(q) := 1$ for $q \in Q_{\text{reset}}$, and $\Phi'_1(q) := 0$ for $q \in Q_{\text{orig}}$. Clearly, the only transitions that affect Φ'_1 are $\langle\mathsf{leader}\rangle$ and the third part of $\langle\mathsf{init\text{-}reset}\rangle$, both of these reducing the potential by 1. It is thus not possible for Φ'_1 to increase. One of these transitions is enabled, so Φ'_1 can decrease at C.

In particular, note that for each $q \in S$, where $S := \{q \in q' : \Phi'_1(q) > 0\}$ are the states with positive potential, there is a transition reducing Φ'_1 using two agents in q. Using T to denote these transitions, we get $\Phi(C) \le 2C(S) \le 2\sum_{t\in T} \text{tmin}_t(C)$ and thus (via Lemma 66), $(\Phi(C) - \Phi(C_{\text{term}}))^2 \le \Phi(C)^2 \le 4|T|\,\text{speed}_T(C)$. Finally, we note $|T| = |Q| + \beta + 1$.

Case 2. $\pi(C)$ is not terminal. In this case, we will argue that the refined transitions of $\mathcal{P}$ are likely to occur. We define $\Phi'_2((q, \pm)) := \Phi(q)$ for $q \in Q$ (recall that Φ is the potential function of $\mathcal{P}$), and set Φ'_2 to 0 elsewhere. Φ'_2 is reduced precisely by the $\langle\mathsf{execute}\rangle$ transitions, and increased only by the third case of $\langle\mathsf{init\text{-}reset}\rangle$.

As we exclude Case 1, we have $C(Q_{\text{supp}} \cup Q_{\text{gate}} \cup Q_{\text{reset}}) = |Q| + \beta + 1$. This implies $\Phi(\pi(C)) = \Phi'_2(C)$ (we even get $\pi(C)(q) = C((q, \pm))$ for $q \in Q$) and ensures that the third case of $\langle\mathsf{init\text{-}reset}\rangle$ cannot be executed by any configuration reachable from C. Having $\pi(C)(q) = C((q, \pm))$ for $q \in Q$ then ensures that for a transition of $\mathcal{P}$ enabled at $\pi(C)$, there is a corresponding $\langle\mathsf{execute}\rangle$ transition enabled at C.

We now want to show that $\pi(C)$ is well-initialised.

$$\pi(C)(I) \overset{(1)}{=} C(I') \overset{(2)}{\le} \tfrac{2}{3}|C| - |H'| = \tfrac{2}{3}|C| - |H| - |Q| - \beta - 1 \overset{(3)}{\le} \tfrac{2}{3}C(Q_{\text{orig}}) - |H|$$

At (1), we use that states in Q have no incoming transitions in $\mathcal{P}$, so $Q \times \{+\}$ have no incoming transitions in $\mathcal{P}'$ and are always empty. (2) follows from C being well-initialised. For (3) we use $C(Q' \setminus Q_{\text{orig}}) = C(Q_{\text{supp}} \cup Q_{\text{gate}} \cup Q_{\text{reset}}) = |Q| + \beta + 1$. Finally, due to $C(Q_{\text{orig}}) = |\pi(C)|$ we derive that $\pi(C)$ is well-initialised.

This allows us to use that Φ is α-rapidly decreasing:

$$(\Phi'_2(C) - \Phi'_2(C_{\text{term}}))^2 = (\Phi(\pi(C)) - \Phi(\pi(C_{\text{term}})))^2 \le \alpha\,\text{speed}(\pi(C))$$

It remains to show $\text{speed}(\pi(C)) \le \text{speed}_T(C)$, where T are the $\langle\mathsf{execute}\rangle$ transitions. For each transition $t \in T$ we have four corresponding transitions $t_1, ..., t_4 \in T$, one for each choice of $\pm$. The bound $\text{tmin}_t(\pi(C)) \le \sum_i \text{tmin}_{t_i}(C)$ then follows from Corollary 64.

Case 3. $C(Q \times \{+\}) > 0$. Here, we show that all "+" flags are eliminated quickly. We set $\Phi'_3((q, +)) = 1$ for $q \in Q$ and 0 elsewhere. We know that $\pi(C)$ is terminal (else we would be in Case 2), and it must remain so. Hence $\langle\mathsf{execute}\rangle$ is disabled and no transition increases Φ'_3. Also, the first case of $\langle\mathsf{init\text{-}reset}\rangle$ is enabled and can reduce the potential.

For every $q \in Q'$ with $\Phi'_3(q) > 0$ we have a $\langle\mathsf{denotify}\rangle$ transition which decreases Φ'_3 and uses only agents in q. Similarly to Φ'_1, we use T to denote the set of these transitions, and find $(\Phi(C) - \Phi(C_{\text{term}}))^2 \le |T|\,\text{speed}_T(C)$, noting $|T| \le |Q|$.

Case 4. $C(\{0,...,|Q|-1\}) > 0$ or $C((q,1)) \neq 1$ for some $q \in \text{supp}(\pi(C))$. In this case, we show that the agents in Q_{supp} stabilise quickly. We use the potential

$$\begin{aligned} \Phi'_4((q,0)) &:= \Phi'_4((q,!)) + 1 := 2 && \text{for } q \in Q \\ \Phi'_4(i) &:= 3(|Q| - i) && \text{for } i = 0,...,|Q| \end{aligned}$$

Again, Φ'_4 is 0 elsewhere. Due to the conditions on Cases 1 and 3, the only transition producing a state $\{0,...,|Q|-1\}$ is the first part of $\langle\textsf{reset}\rangle$, which decreases Φ'_4. Otherwise, state $Q \times \{0\}$ cannot be produced. The only other transitions affecting the potential are $\langle\textsf{detect}\rangle$ and the second case of $\langle\textsf{init-reset}\rangle$, which both decrease Φ'_4. One of the above transitions, which we again denote by T, is always enabled, so $\text{speed}_T(C) \geq 1$. Additionally, we have $C(Q_{\text{supp}}) = |Q|$ and $C(Q_{\text{reset}}) = 1$, so $\Phi'_4(C) \leq 5|Q| \leq 5|Q|\,\text{speed}_T(C)$.

Case 5. C is not terminal. Finally, we consider the speed at which gates stabilise and the computer terminates.

$$\begin{aligned} \Phi'_5((g,b_1,b_2,b_3)) &:= \mathbf{1}_\perp(b_2) + \mathbf{1}_\perp(b_3) && \text{for } g \in G \\ \Phi'_5(|Q| + i) &:= 3(\beta - i) && \text{for } i = 0,...,\beta \end{aligned}$$

where $\mathbf{1}_\perp(\perp) := 1$ and $\mathbf{1}_\perp(0) := \mathbf{1}_\perp(1) := 0$. At this point, only transitions $\langle\textsf{gate}\rangle$ and the second part of $\langle\textsf{reset}\rangle$ are active, and both reduce Φ'_5. We denote them by T and, analogous to Phase 4, we get the estimate $\Phi'_5(C) \leq 5\beta\,\text{speed}_T(C)$ and find that one of these transitions is always enabled. ◀

E Conversion 3: Removing Helpers

In this section, we show to obtain fast succinct population computers without helpers for all predicates $\varphi \in \mathit{QFPA}$. For this we prove a general conversion theorem, however the theorem does not convert a computer into an equivalent one. Instead it starts with a computer for a predicate we call $\tau(\varphi)$ to generate a computer for φ. In this predicate, one input stands for two inputs of the original predicate. For example, for majority $x - y \geq 0$ we would have $x_1 + 2x_2 - y_1 - 2y_2 \geq 0$.

Formally, given predicate $\varphi : \mathbb{N}^I \to \{0, 1\}$, we define a new set of inputs $I' := \{x' : x \in I\}$ consisting of a copy of I, and define $\tau(\varphi) : \mathbb{N}^{I \cup I'} \to \{0, 1\}$ as

$$\tau(\varphi)(x_1, ..., x_k, x'_1, ..., x'_k) := \varphi(x_1 + 2x'_1, ..., x_k + 2x'_k)$$

for all inputs $x_1, ..., x_k, x'_1, ..., x'_k \in \mathbb{N}$. Clearly, $\tau(\varphi) \in \mathit{QFPA}$ if $\varphi \in \mathit{QFPA}$ and $|\tau(\varphi)| \in \mathcal{O}(|\varphi|)$, since we can obtain it by replacing every occurrence of x_i in φ by $x_i + 2x'_i$.

The conversion theorem is as follows. As for the previous conversions, it contains a part giving a bound on the speed of the resulting computer, which is independent from the rest and relies on the definitions summarised in Appendix H, in particular the definition of potential groups.

▶ **Theorem 46.** *Let $\varphi \in \mathit{QFPA}$ and let $\mathcal{P} = (Q, \delta, I, O, H)$ be a bounded binary population computer with marked consensus output deciding $\tau(\varphi)$, s.t. states in I have no incoming transitions and every configuration in $\mathbb{N}^I$ is terminal.*

Then there exists a bounded binary population computer $\mathcal{P}' = (Q', \delta', I', O', \emptyset)$ of size $\mathcal{O}(\text{size}_2(\mathcal{P}))$ without helpers and with a marked consensus output deciding φ for inputs of size at least $|I| + 2|H|$.

Additionally, if $\mathcal{P}$ has a potential group of size e, which is α-rapidly decreasing in all well-initialised configurations, then $\mathcal{P}'$ has a potential group of size $e + 1$, which is $(\alpha + (|I| + |H|)|H|^2)$-rapidly decreasing in all reachable configurations of size at least $6|I| + 10|H|$.

It will be proven in the following subsections, as Lemma 49 and Lemma 50, where the latter contains the speed bound.

E.1 Construction

Intuitively, two agents in state $x \in I$ can combine to an agent in x', freeing one agent to act as helper. It is important to ensure that the protocol does not run out of inputs until we have generated enough helpers to ensure correctness, so we only distribute helpers once we have collected $|H|$ of them. Condition 2 ensures that the protocol cannot execute transitions before this point.

Ideally, we would add a state h and transitions

$$x, x \mapsto 2x, \mathsf{h} \qquad \text{for } x \in I \qquad \langle\mathsf{double}\rangle$$

$$|H| \cdot \mathsf{h} \mapsto H \qquad \langle\mathsf{helper}\rangle$$

However, our resulting model cannot use multiway transitions. (For technical reasons it is easier to remove multiways before helpers.) Instead of $\langle\mathsf{helper}\rangle$, we will therefore inline the construction from Section 7.1 (simplified slightly).

We add states $Q_{\text{helper}} := \{\triangle_i, \triangledown_i : i = 0, ..., |H|\}$, and identify $\triangle_1$ with h and $\triangledown_1$ with h_1; here, $h_1, ..., h_m$ is an enumeration of H. For $i, j \in \{1, ..., |H| - 1\}$ we have transitions

$$\begin{aligned} \triangle_i, \triangle_j &\mapsto \triangle_{i+j}, \triangle_0 && \text{if } i + j < |H| \\ \triangle_i, \triangle_j &\mapsto \triangledown_{|H|}, \triangle_{i+j-|H|} && \text{if } i + j \geq |H| \qquad \langle\mathsf{helper}\rangle \\ \triangledown_{i+1}, \triangle_0 &\mapsto \triangledown_i, h_{i+1} \end{aligned}$$

Of course, we also let the computer $\mathcal{P}$ execute normally, so we add all transitions of δ to δ'. Finally, we choose $O'(S) := O(S \cap Q)$ for all $S \subseteq Q'$ (note that $Q \subseteq Q'$).

E.2 Correctness

As for the other conversions, we define a linear map $\pi : \mathbb{N}^{Q'} \to \mathbb{N}^Q$ to translate configurations of $\mathcal{P}'$ to ones of $\mathcal{P}$. Here, we simply choose $\pi(C)(q) := C(q)$ for $q \in Q$. We do not, however, show that $\mathcal{P}'$ refines $\mathcal{P}$, as that does not hold: transitions $\langle\mathsf{double}\rangle$ and $\langle\mathsf{helper}\rangle$ change $\pi(C)$ in a way that is not compatible with an execution of $\mathcal{P}$.

Instead, we start by showing that it suffices to consider only certain transition sequences, where the $\langle\mathsf{double}\rangle$ and $\langle\mathsf{helper}\rangle$ transitions occur only in the beginning. After that point, our proof proceeds just as for the refinement results.

▶ **Definition 47.** *Let $\sigma_1, \sigma_2, \sigma_3 \in (\delta')^*$ denote (finite) sequences of transitions, where σ_1 contains only $\langle\mathsf{double}\rangle$ transitions, σ_2 only $\langle\mathsf{helper}\rangle$ transitions, and $\sigma_3 \in \delta^*$. The sequence $\sigma := \sigma_1\sigma_2\sigma_3$ is then called* good.

▶ **Lemma 48.** *If $C \in \mathbb{N}^{Q'}$ is reachable from an initial configuration $C_0 \in \mathbb{N}^I$, then there is a good sequence $\sigma \in (\delta')^*$ with $C_0 \to_\sigma C$.*

Proof. States $x \in I$ have no incoming transitions (condition 1 of Theorem 46), so the number of agents in x is monotonically decreasing during a run. Hence a $\langle\mathsf{double}\rangle$ transition can always be moved to any earlier position in a transition sequence.

Similarly, the states used as input by a $\langle\mathsf{helper}\rangle$ transition only have incoming $\langle\mathsf{double}\rangle$ or $\langle\mathsf{helper}\rangle$ transitions, so if a $\langle\mathsf{helper}\rangle$ transition is preceded by a δ transition, their order may be swapped. ◀

▶ **Lemma 49.** *$\mathcal{P}'$ is bounded and decides φ for inputs of size at least $2|H| + |I|$.*

Proof. Let $C \in \mathbb{N}^{Q'}$ denote a terminal configuration reachable from an input configuration C_0 with at least $2|H| + |Q|$ agents. Lemma 48 then implies that there are configurations C_1, C_2 with $C_0 \to C_1 \to C_2 \to C$, s.t. going from C_0 to C_1 executes only $\langle\mathsf{double}\rangle$ transitions, going from C_1 to C_2 only $\langle\mathsf{helper}\rangle$ transitions, and from C_2 to C only transitions in δ. We now consider two cases.

If $\langle\mathsf{double}\rangle$ is not enabled at C_1, then $C_1(\mathsf{h}) \geq |H|$, as there can be at most $|I|$ agents left in $C_1(I)$. It is not possible to remove agents from Q_{helper} without executing $\langle\mathsf{helper}\rangle$, so at some point at least $|H|$ agents in $C_1(\mathsf{h})$ will be distributed to $h_1, ..., h_{|H|}$ and we get $C_2 \geq H$

Else, $\langle\mathsf{double}\rangle$ is enabled at C_1 (but not at C), thus some transition removing agents from $C(I)$ must have occurred between C_2 and C (as $\langle\mathsf{helper}\rangle$ transitions cannot do so). Due to condition 2 of Theorem 46, we get $C_2(\text{supp}(H)) > 0$, which, due to the construction of transition $\langle\mathsf{helper}\rangle$, implies $C_2 \geq H$. (In particular, helpers are distributed in batches of H.)

So in both cases we have $C_2 \geq H$ and therefore find that $\pi(C_2)$ is an initial configuration (of $\mathcal{P}$). Between C_0 and C_2, only transition $\langle\mathsf{double}\rangle$ affects states $I \cup I'$, and it preserves the value of double(φ). (Also note $\varphi(C_0) = \text{double}(\varphi)(C_0)$, as only agents in I are present.)

As C can be reached from C_2 by executing only transitions in δ, we also get that $\pi(C)$ is a reachable configuration of $\mathcal{P}$. Moreover, C is terminal w.r.t. $\delta' \supseteq \delta$, so $\pi(C)$ is a terminal configuration of C. We have defined O' s.t. $O'(C) = O(\pi(C))$ and thus get the correct output.

To argue that $\mathcal{P}'$ is bounded, we note that $\delta' \setminus \delta$ is acyclic. So if $\mathcal{P}'$ has arbitrarily long runs, then, due to Lemma 48, $\mathcal{P}$ has as well. But that cannot be, as $\mathcal{P}$ is bounded. ◀

E.3 Speed

As mentioned at the start of this section, the definitions for speed analysis can be found in section H.

▶ **Lemma 50.** *If a potential group Φ for $\mathcal{P}$ of size e exists, which is α-rapidly decreasing in all well-initialised configurations, then there is a potential group Φ' for $\mathcal{P}'$ of size $e+1$, which is $(\alpha + (2|H| + |I|)|H|^2)$-rapidly decreasing in all reachable configurations of size at least $6|H| + 10|I|$.*

Proof. Let $(\Phi_1, ..., \Phi_e) := \Phi$. We define $\Phi' := (\Phi'_1, \Phi_1, \Phi_2, ..., \Phi_e)$, where:

$$\begin{aligned} \Phi'_1(q) &:= 2 && \text{for } q \in I \\ \Phi'_1(\triangle_i) &:= i+1 && \text{for } i \in \{1, ..., |H|-1\} \\ \Phi'_1(\triangledown_i) &:= i && \text{for } i \in \{2, ..., |H|\} \end{aligned}$$

As usual, other states have potential 0, and we extend Φ_i to $\mathbb{N}^{Q'}$ by setting the weight of states in $Q' \setminus Q$ to 0.

Clearly, transitions $\langle\mathsf{double}\rangle$ and $\langle\mathsf{helper}\rangle$ decrease Φ'_1, while transitions in δ cannot increase it due to condition 1 of Theorem 46.

Now, let C denote a configuration reachable from an input of size at least $2|H| + |I|$. To show that Φ' is rapidly decreasing in C, we differentiate between two cases.

Case 1. If either $\langle\mathsf{double}\rangle$ or $\langle\mathsf{helper}\rangle$ is enabled at C, we show that Φ'_1 is active. It has already been shown that Φ'_1 cannot decrease. To show that Φ'_1 is rapidly-decreasing, let $S := \{q \in Q' : \Phi'_1(q) > 0\}$ denote the states with positive potential. For each state $q \in I \cup \{\triangle_1, ..., \triangle_{|H|-1}\} \subseteq S$ we have a transition using only agents in q. For each other state, i.e. $q = \triangledown_i \in S$ for some i, we observe $C(\triangle_0) \geq C(\triangledown_i)$, as the construction guarantees that enough agents in $\triangle_0$ exist. So in total we have $\Phi(C) \leq |H|C(S)$ and $C(S)^2 \leq |S| \operatorname{speed}_{\delta_>}(C)$ by Lemma 66. Using $|S| \leq |I| + 2|H|$, Φ'_1 is $(|I| + 2|H|)|H|^2$-rapidly decreasing in C.

Case 2. Otherwise, $C(I) \leq |I|$, $C(Q_{\text{helper}}) < |H|$ and, due to our construction, $C(I') \leq |C|/2$. From the second, we derive $C(Q) > |C| - |H|$, which we combine with the other two to get $C(I \cup I') \leq |I| + |C|/2 < |I| + (C(Q) + |H|)/2$. Rearranging terms yields $C(I \cup I') + |H| \leq C(Q)/2 + |I| + \frac{3}{2}|H|$.

Now, we use $|C| \geq 6|I| + 10|H|$ to get $C(Q) > |C| - |H| \geq 6|I| + 9|H|$, so $C(I \cup I') + |H| \leq |I| + \frac{3}{2}|H| + C(Q)/2 \leq \frac{2}{3}C(Q)$. Noting $C(Q) = |\pi(C)|$, we find that $\pi(C)$ is well-initialised, so Φ is α-rapidly decreasing in $\pi(C)$. This extends directly to Φ'. ◀

F Conversion 4: Fast Output Broadcast

This section describes the final step of our conversions, which shows how to construct an equivalent population protocol out of a given bounded binary population computer $\mathcal{P}$ without helpers using a marked consensus output. We show the following general conversion result. The first half of the theorem can be read independently of the second half, which uses definitions introduced exclusively for the purpose of speed analysis. These are summarised in Section H.

▶ **Theorem 51.** *Let $\mathcal{P} = (Q, \delta, I, O, \emptyset)$ be a bounded binary population computer without helpers and using a marked consensus output deciding a predicate φ for inputs of size at least m. Then there exists a terminating population protocol $\mathcal{P}'$ with $4|Q|$ states deciding φ for inputs of size at least m. Additionally:*

- *If $\mathcal{P}$ has a potential function, then $\mathcal{P}'$ terminates in $\mathcal{O}(n^3)$ interactions in expectation.*
- *If $\mathcal{P}$ has a potential group of size e, which is α-rapidly decreasing in all reachable configurations of size at least m, then $\mathcal{P}'$ terminates in $\mathcal{O}(e(\sqrt{\alpha}|Q| + \alpha)n^2)$ interactions in expectation.*

As usual, the proof of Theorem 51 will span the remainder of this section and consist of multiple results, in particular lemmata 53 and 55.

F.1 Construction

The obvious approach would be to add an extra bit to every state, which will be the opinion of the agent, and is set whenever the agent meets a marked agent. However, this would be slightly too slow: convincing the i-th agent takes roughly $n^2/(n-i)$ steps (the inverse of the probability that the single marked agent meets one of the remaining $n-i$ agents with the wrong opinion). In total, this sums to $n^2 \log n$ steps.

Instead, we are going to modify the procedure slightly. Whenever an agent meets a marked agent, besides assuming the correct opinion it will receive a token, which it can use once, to convince another agent.

Formally, let $Q_0, Q_1 \subseteq Q$ denote the states defining the marked consensus output O, and set $Q_\perp := Q \setminus (Q_0 \cup Q_1)$. We use states $Q' := Q \times \{0,1\}^2$, where the second component denotes the current opinion of the agent, and the third whether it has a token.

Let $q, p \in Q$. We want to execute $\mathcal{P}$ simultaneously to the following transitions. To write this down, we choose q', p' as the result of a transition $(q, p \mapsto q', p') \in \delta$, if such a transition exists, else we set $q' := q, p' := p$.

For convenience, we write $*$ if the component does not matter. A $*$ in the result of the transition indicates that this component is left unchanged. Based on our definitions the agents of a transitions have no order, so $(q, p \mapsto q', p')$ and $(p, q \mapsto q', p')$ are the same transition. Let $i \in \{0, 1\}$.

If an agent meets a marked agent, then the former will assume the latter's opinion and receive a token.

$$(q, *, *), (p, *, *) \mapsto (q', i, 1), (p', i, 1) \qquad \text{if } \{q', p'\} \cap Q_i \neq \emptyset \qquad \langle\mathsf{certify}\rangle$$

If an agent with token meets a non-token agent with opposing opinion, the latter is convinced and the token consumed. Similarly, if two tokens held by agents with opposing opinions meet, the tokens are simply dropped.

$$(q, i, 1), (p, 1-i, 0) \mapsto (q', i, 0), (p', i, 0) \qquad \langle\textsf{convince}\rangle$$
$$(q, i, 1), (p, 1-i, 1) \mapsto (q', i, 0), (p', 1-i, 0) \qquad \langle\textsf{drop}\rangle$$

Otherwise, nothing happens.

$$(q, *, *), (p, *, *) \mapsto (q', *, *), (p', *, *) \qquad \text{if } (q, p) \neq (q', p') \qquad \langle\textsf{noop}\rangle$$

As defined above, our transitions are not deterministic. If multiple transitions are possible, we will pick a $\langle\textsf{certify}\rangle$ transition, or (if there are none) a $\langle\textsf{convince}\rangle$ or $\langle\textsf{drop}\rangle$ transition.

Finally, we choose O' as the consensus output given by the partition $O'_i := Q \times \{i\} \times \{0, 1\}$, for $i \in \{0, 1\}$ and $I' := I$, where we identify $q \in I$ with $(q, 0, 0)$.

F.2 Correctness

As for the other conversions, let $\pi : \mathbb{N}^{Q'} \to \mathbb{N}^Q$ denote a mapping from configurations of $\mathcal{P}'$ to ones of $\mathcal{P}$. We choose $\pi(C)(q) := C(q \times \{0, 1\}^2)$, so π simply projects onto the first component. Again, we are going to show that $\mathcal{P}'$ refines $\mathcal{P}$, as defined in Definition 61.

▶ **Lemma 52.** *$\mathcal{P}'$ refines $\mathcal{P}$.*

Proof. Properties 1 and 2 follow immediately from the construction. For the third property, let C denote a reachable, terminal configuration of $\mathcal{P}'$. If a transition of $\mathcal{P}$ were enabled at $\pi(C)$, the corresponding $\langle\textsf{noop}\rangle$ transition would be enabled at C, so $\pi(C)$ must be terminal as well.

As $\mathcal{P}$ has a marked consensus output, there is an agent in a state $q \in Q_i$, where $i \in \{0, 1\}$ is the output of $\mathcal{P}$ and $\pi(C)(q) > 0$. (Recall that $Q_0, Q_1 \subseteq Q$ are the states inducing the output function of $\mathcal{P}$.) By the definition of π this implies that there is a $q' \in \{q\} \times \{0, 1\}^2$ with $C(q') > 0$.

We now claim $\mathrm{supp}(C) \subseteq Q'_i$, so assume the contrary and pick a $p' \in Q' \setminus Q'_{1-i}$ with $C(p') > 0$. But agents q', p' can now execute transition $\langle\textsf{certify}\rangle$, which contradicts C being terminal. Thus the claim is shown and our choice of O' yields $O'(\mathrm{supp}(C)) = i$. ◀

In contrast to the other conversions, $\mathcal{P}'$ is not bounded. It is, however, still terminating, which is sufficient to rely on the refinement property to show that $\mathcal{P}'$ and $\mathcal{P}$ decide the same predicate.

▶ **Lemma 53.** *$\mathcal{P}'$ is terminating and decides φ for inputs of size at least m.*

Proof. Let C denote a reachable configuration of $\mathcal{P}'$. As $\mathcal{P}'$ refines $\mathcal{P}$, $\pi(C)$ is reachable in $\mathcal{P}$; as $\mathcal{P}$ is bounded $\pi(C)$ can reach some terminal configuration D. We can execute a corresponding sequence of transitions in $\mathcal{P}'$ and find a configuration C' with $C \to C'$ and $\pi(C')$ terminal. At this point, $\pi(C)(Q_i) > 0$ and $\pi(C)(Q_{1-i}) = 0$ for an $i \in \{0, 1\}$, which corresponds to the output of $\mathcal{P}$. By executing at most $n - 1$ $\langle\textsf{certify}\rangle$ transitions, we can reach a configuration where all agents have opinion i and a token; the resulting configuration is terminal.

To summarise, we have argued that any reachable configuration can reach a terminal configuration. Hence any infinite run can reach terminal configurations infinitely often and any fair run will eventually terminate.

To show that $\mathcal{P}'$ decides φ for inputs of size at least m we would like to simply cite Lemma 62, relying on the notion of refinement. Formally, however, this does not work, as we would need to assume that $\mathcal{P}$ decides φ for *all* inputs. Fortunately, the proof can trivially be adapted to consider only inputs of size at least m. ◀

F.3 Speed

As mentioned at the start of this section, the definitions and results for speed analysis can be found in Section H. In particular, it contains Proposition 60, which shows that any population protocol with a potential group reaches a terminal configuration within a quadratic number of steps.

This can be applied to $\mathcal{P}$ (i.e. the protocol we are converting to $\mathcal{P}'$) by ignoring its output, to show that the refined protocol quickly terminates. Afterwards, we have to prove that we can broadcast the result to all agents in a reasonable amount of time.

Intuitively, the broadcast operates in two phases. First, the marked agents ensures that “many”, e.g. $\frac{2}{3}n$, agents have both the correct opinion and a token. Afterwards, these agents use their tokens to convince the remaining agents of the correct opinion. The first phase is fast as there are always linearly many agents for the marked agent to interact with, and in the second phase the remaining agents with wrong opinion always have linearly many agents with tokens to interact with.

▶ **Lemma 54.** *If $\mathcal{P}'$ reaches a configuration C with $\pi(C)$ terminal within $f(n)$ random interactions in expectation, $\mathcal{P}'$ stabilises after $\mathcal{O}(f(n)+n^2)$ random interactions in expectation.*

Proof. Let C_1 denote a configuration s.t. $\pi(C_1)$ is terminal. In C_1, we have at least one marked agent for the correct answer $b := O(\text{supp}(C_1))$, and no marked agents for the wrong answer, and this will not change during the remainder of the computation.

Let $f_{ij} := C(Q \times \{i\} \times \{j\})$, for $i, j \in \{0, 1\}$ denote the number of agents with opinion i and holding j tokens. Then $F := f_{b1} - f_{(1-b)0} - 2f_{(1-b)1}$ counts the agents with correct opinion and a token, subtracting both the agents with the wrong opinion and the tokens held by agents with the wrong opinion. It is easy to see that this number cannot decrease, and will increase whenever a marked agent meets an agent that either has the wrong opinion, or does not have a token.

If $F \leq \frac{2}{5}n$, we have $f_{b1} \leq \frac{4}{5}n$, so in expectation we have to wait $5n$ steps for F to increase. As $F \geq -2n$ by definition, we need $5n(\frac{2}{5}n + 2n) \in \mathcal{O}(n^2)$ steps until we have $F \geq \frac{2}{5}n$.

As noted, F cannot decrease, so after that point we always have $f_{b1} \geq F \geq \frac{2}{5}n$. Whenever an agent with opinion $1 - b$ meets an agent in f_{b1}, the value $f_{(1-b)0} + 2f_{(1-b)1}$ decreases. As long as agents with the wrong opinion exist, this has to happen after at most $\frac{5}{2}n$ steps in expectation. Noting $f_{(1-b)0} + 2f_{(1-b)1} \leq 2n$ we find that after $\mathcal{O}(n^2)$ steps no agents with opinion $1 - b$ remain. ◀

▶ **Lemma 55.** **(a)** *If there is a potential function Φ for $\mathcal{P}$, then $\mathcal{P}'$ stabilises after $\mathcal{O}(n^3)$ random interactions in expectation.*

(b) *Let Φ denote a potential group for $\mathcal{P}$ of size e which is rapidly decreasing in all reachable configurations with at least m agents. Then $\mathcal{P}'$ stabilises after $\mathcal{O}(e(\sqrt{\alpha}|Q| + \alpha)\, n^2)$ interactions.*

Proof. In both cases we apply Lemma 54. For (a) we simply note $f(n) \in \mathcal{O}(n^3)$ based on the existence of a potential function Φ: at every step at least one transition can make progress, so every $\mathcal{O}(n^2)$ steps the potential reduces by one, and an initial configuration C_0 has $\Phi(C_0) \in \mathcal{O}(n)$. For (b) we instead use Proposition 60. ◀

G A General Conversion Theorem

Our paper makes two statements about conversion of population computers: a general conversion theorem for bounded population computers, and a specialised version that only applies to rapid computers, but provides better bounds. Here we describe the former, using the general conversion theorems of the previous sections.

Before going into the conversions, in Section G.1 we take a brief detour and show that bounded population computers have a potential function. In particular, they are linearly bounded. When analysing the running time, this will become crucial to show that the resulting population protocols stabilise within $\mathcal{O}(n^3)$ interactions.

Afterwards, Section G.2 will assemble the necessary components to complete the proof.

G.1 Boundedness and Potential Functions

In this section we are going to prove that boundedness and existence of a potential function are equivalent, as long as the population computer has no "useless" states.

Let $\mathcal{P} = (Q, \delta, I, O, H)$ denote some population computer. Our goal is to prove the following result.

▶ **Lemma 11.** *If $\mathcal{P}$ has a reachable configuration C_q with $C_q(q) > 0$ for each $q \in Q$, then $\mathcal{P}$ is bounded iff there is a potential function for $\mathcal{P}$.*

For the remainder of this section we are going to assume that all states of $\mathcal{P}$ are reachable, i.e. for every $q \in Q$ there is a reachable configuration C with $C(q) > 0$. As a simple consequence we can show that is is possible to put arbitrarily many agents into every state.

▶ **Lemma 56.** *Let $i \in \mathbb{N}$. There is a reachable configuration C with $C(q) \geq i$ for $q \in Q$.*

Proof. Transitions of population computers (and population protocols) are "monotonic", so for any $C \to C'$ and $D \to D'$ we have $C + D \to C' + D'$. Additionally, for two initial configurations C_0, D_0 the configuration $C_0 + D_0$ is initial as well. (Recall that for helpers, as opposed to leaders, it is allowed to provide more agents then are required.) Combining these two facts we find that $C + D$ is reachable, for any reachable configurations C, D. As we require that for every $q \in Q$ there is some reachable C s.t. $C(q) > 0$, the desired statement follows. ◀

The proof invokes Farkas' lemma; we thus need to convert a statement about boundedness and potential functions to linear algebra. To this end, we define the *incidence matrix* of $\mathcal{P}$ as $A \in \mathbb{Z}^{\delta \times Q}$ s.t. the t-th row is $A_t := s - r$, for $t = (r \mapsto s) \in \delta$. Intuitively, given a vector $y \in \mathbb{N}^\delta$ which assigns each transition a count, $A^\top y$ is the change in the number of agents of each state after executing the transitions of y.

▶ **Lemma 57.** *The following are equivalent:*

(a) *$\mathcal{P}$ is bounded.*
(b) *$A^\top y \neq 0$ for all $y \in \mathbb{N}^\delta$ with $y \neq 0$.*
(c) *$A^\top y \neq 0$ for all $y \in \mathbb{R}^\delta_{\geq 0}$ with $y \neq 0$.*
(d) *$Ax \leq -1$ for some $x \in \mathbb{R}^Q$.*
(e) *There is a potential function Φ for $\mathcal{P}$.*

Proof. "$(a) \Rightarrow (b)$": Assume that (b) does not hold, so there is a nonempty multiset $y \in \mathbb{N}^\delta$ with $A^\top y = 0$. Let $t_1, ..., t_k \in \delta$ denote an enumeration of y. Due to the definition of A, $A^\top y = 0$ means that executing the sequence $t_1 t_2 ... t_k$ has no effect. Formally, for any C, C'

with $C \to_{t_1} \dots \to_{t_k} C'$ we get $C = C'$. It suffices to find such a configuration C which is reachable; as then we can clearly construct an infinite run, contradicting (a).

Using Lemma 56, we pick a C with $C(q) \geq km$, where $m := \max\{|r| : (r \mapsto s) \in \delta\}$ is the maximum size of any transition. A single transition moves at most m agents, so the sequence $t_1, \dots, t_k$ can be executed at C.

"$(b) \Rightarrow (c)$": We argue by contraposition and assume that $\{y \in \mathbb{R}^{\delta}_{\geq 0} \setminus \{0\} : A^\top y = 0\}$ is not empty. Then there is some $\varepsilon \in \mathbb{Q}$ with $\varepsilon > 0$ s.t. $\{y \in \mathbb{R}^{\delta}_{\geq 0} : A^\top y = 0 \wedge \mathbf{1}^\top y \geq \varepsilon\}$ is not empty either. This is a satisfiable system of linear inequalities and thus has a rational solution $y^* \in \mathbb{Q}^\delta$. As $y^* \geq 0$ we can find a $\mu > 0$ with $\mu y^* \in \mathbb{N}^\delta$, showing the negation of (b).

"$(c) \Rightarrow (d)$": Due to $y \geq 0$ the condition $y \neq 0$ is equivalent to $\mathbf{1}^\top y > 0$; using $\mathbf{1}$ for the all-ones vector of appropriate dimension. In other words, the system $\{y \in \mathbb{R}^{\delta}_{\geq 0} : A^\top y = 0 \wedge -\mathbf{1}^\top y < 0\}$ has no solution. We apply Farkas' lemma; this shows that the system $\{x \in \mathbb{R}^Q : Ax \leq -1\}$ does have a solution.

"$(d) \Rightarrow (e)$": Again, $Ax \leq -1$ is a system of linear inequalities, so if it has a solution it has a rational solution as well. Scaling a solution with a factor $\mu > 1$ yields another solution, so there is also an integer solution. Let $x \in \mathbb{Z}^Q$ with $Ax \leq -1$ denote such a solution. The weights for our potential function have to be natural numbers, so let $x_{\min} := \min_{q \in Q} x(q)$ and set $x^* := x - x_{\min}\mathbf{1}$. Finally define m as above and choose the weights $w := mx^*$.

It remains to show that the linear function Φ induced by w is actually a potential function. Let $t = (r \mapsto s) \in \delta$ be a transition. We have $\Phi(r) - \Phi(s) = \sum_q (r(q) - s(q))w(q) = -A_t^\top w$, with A_t the t-th row of A, as above. Furthermore, we get $A_t^\top w = mA_t^\top(x - x_{\min}\mathbf{1})$. Using $|r| = |s|$ we get $A_t^\top \mathbf{1} = 0$; and $Ax \leq -1$ implies $A_t^\top x \leq -1$. In total we thus have $\Phi(r) - \Phi(s) \geq m \geq |r| - 1$.

"$(e) \Rightarrow (a)$": For any initial configuration C_0 we have $\Phi(C_0) \in \mathcal{O}(n)$ as Φ is a linear function. Since, by definition, $\Phi(C) \geq 0$ for all configurations C and any transition strictly reduces Φ, a run starting at C_0 can execute at most $\Phi(C_0)$ transitions. ◀

G.2 Proof of Theorem 2

We now prove the following result.

▶ **Theorem 2.** *Every bounded population computer of size m deciding* double(φ) *can be converted into a terminating population protocol with $\mathcal{O}(m^2)$ states which decides φ in at most $\mathcal{O}(f(m)\, n^3)$ interactions for inputs of size $\Omega(m)$, for some function f.*

Let $\mathcal{P} = (Q, \delta, I, O, H)$ denote a bounded population computer of size m deciding a predicate double(φ). We want to apply theorems 27, 39, 46 and 51 in sequence. For this, however, we need the two additional conditions required by Theorem 46:

- states in I have no incoming transitions, and
- every configuration in $\mathbb{N}^I$ is terminal.

We can easily modify $\mathcal{P}$ to ensure that they hold. We define a new computer $\mathcal{P}' = (Q', \delta', I', O', H')$ as follows:

- $Q' := Q \cup \{h\} \cup \{q_* : q \in I\}$,
- $\delta' := \delta \cup \{q_*, h \mapsto q, h : q \in I\}$,
- $I' := \{q_* : q \in I\}$,
- $O'(S) := O(S \cap Q)$, for $S \subseteq Q'$, and
- $H'(q) := H(q)$ for $q \in Q$ and $H(h) := 1$.

We remark that the technical conditions are still necessary for the improved running-time analysis using rapid computers, as these conversions affect their speed. For our purposes we only need that $\mathcal{P}'$ is bounded, which is clearly the case.

The aforementioned conversion theorems can now be applied, resulting in the intermediate computers $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and the population protocol $\mathcal{P}_4$. Using Q_i to denote the states of $\mathcal{P}_i$, we get

$$|Q_4| = 4|Q_3| \in \mathcal{O}(\text{size}_2(\mathcal{P}_2)) \in \mathcal{O}(\text{size}(\mathcal{P}_1) \cdot |Q_1|) \subseteq \mathcal{O}(m^2)$$

For the time bound we use that $\mathcal{P}_3$ is bounded, so by Lemma 11 there is a potential function Φ for $\mathcal{P}_3$ and Theorem 51 guarantees stabilisation within $\mathcal{O}(n^3)$ interactions.

H Potential Groups

In this section we describe how we generalise the notion of potential function (described in Section 7.4) to better analyse our conversions (in particular the simulation of output functions and the simulation of helpers). We start by briefly recalling the main definitions of Sections 7.4 and 8.

▶ **Definition 10.** *A function* $\Phi : \mathbb{N}^Q \to \mathbb{N}$ *is* linear *if there exist weights* $w : Q \to \mathbb{N}$ *s.t.* $\Phi(C) = \sum_{q \in Q} w(q)C(q)$ *for every* $C \in \mathbb{N}^Q$*. We write* $\Phi(q)$ *instead of* $w(q)$*. A* potential function (for $\mathcal{P}$) *is a linear function* Φ *such that* $\Phi(r) \geq \Phi(s) + |r| - 1$ *for all* $(r \mapsto s) \in \delta$*.*

▶ **Definition 12.** *Given a configuration* $C \in \mathbb{N}^Q$ *and some transition* $t = (r \mapsto s) \in \delta$*, we let* $\mathrm{tmin}_t(C) := \min\{C(q) : q \in \mathrm{supp}(r)\}$*. For a set of transitions* $T \subseteq \delta$*, we define* $\mathrm{speed}_T(C) := \sum_{t \in T} \mathrm{tmin}_t(C)^2$*, and write* $\mathrm{speed}(C) := \mathrm{speed}_\delta(C)$ *for convenience.*

▶ **Definition 13.** *Let* Φ *denote a potential function for* $\mathcal{P}$ *and let* $\alpha \geq 1$*. We say that* Φ *is* α-rapidly decreasing *at a configuration* C *if* $\mathrm{speed}(C) \geq (\Phi(C) - \Phi(C_{\mathrm{term}}))^2/\alpha$ *for all terminal configurations* C_{term} *with* $C \to C_{\mathrm{term}}$*.*

▶ **Definition 14.** $C \in \mathbb{N}^Q$ *is* well-initialised *if* C *is reachable and* $C(I) + |H| \leq \frac{2}{3}n$*.*

While the above definitions can be applied to all of our conversions, they would lead to large constants in the final speed. These are merely the result of a loose analysis – they do not reflect the actual speed of our protocols.

Mainly, this is due to a single potential function being unable to model computations that consist of multiple phases efficiently. A concrete explanation of this problem in the context of the output conversion is given in Section D.3. In this section we will introduce the formal machinery necessary to better adapt our technique the those constructions, leading to better constants and easier proofs.

We start by extending the definition of rapidly-decreasing to handle linear functions which are not potentials. Here, we do not need to deal with multiway transitions, so let $\mathcal{P} = (Q, \delta, I, H, O)$ denote a binary population computer.

▶ **Definition 58.** *Let* $\Phi : \mathbb{N}^Q \to \mathbb{N}$ *be linear, set* $\delta_> := \{(r \mapsto s) \in \delta : \Phi(r) > \Phi(s)\}$ *to the transitions decreasing* Φ*, and let* $\alpha > 0$*. If* $\mathrm{speed}_{\delta_>}(C) \geq (\Phi(C) - \Phi(C_{\mathrm{term}}))^2/\alpha$ *for a configuration* C *and all terminal configurations* C_{term} *with* $C \to C_{\mathrm{term}}$*, we say that* Φ *is* α-rapidly decreasing in C*.*

The only change compared to Definition 13 is that the speed considers only transitions which reduce the given linear function. For a potential function Φ we have $\delta_> = \delta$ and $\Phi(C) > \Phi(C_{\mathrm{term}})$ for all non-terminal configurations C, making this definition coincide with Definition 13.

To model phases, the general idea is that we construct a family of linear functions $\Phi_1, ..., \Phi_e$. For each configuration C, one of these will be rapidly decreasing (we refer to it as "active"). That alone would not be enough to guarantee linear running time (or any time bound at all), as it would not prevent the other functions from increasing their value. So we require the stronger property that a Φ_i cannot increase once it has been active. We also need that Φ_i can decrease at C, which certifies that some progress can be made. Otherwise, Φ_i might be "rapidly decreasing" but already at its lowest point.

▶ **Definition 59.** *A tuple* $\Phi = (\Phi_1, ..., \Phi_e)$ *where* $\Phi_1, ..., \Phi_e : \mathbb{N}^Q \to \mathbb{N}$ *denote linear maps, is called* potential group (of size e)*.*

*We say that Φ is α-*rapidly decreasing *in C, for $\alpha \geq 1$ and $C \in \mathbb{N}^Q$, if C is terminal or there is some $i \in \{1, ..., e\}$ s.t. Φ_i is α-rapidly decreasing in C, some transition reducing Φ_i is enabled at C, and no transition increasing Φ_i can be executed at any configuration reachable from C. We then call Φ_i* active *at C.*

The definition places no restrictions on the order in which the Φ_i are listed. However, in our proofs we will generally order them in the same fashion as they would become active in a run. Further, our potential groups have the additional property that they decrease lexicographically with each transition.

To close out the section, we show that the above notion does actually lead to a strong speed bound when applied to population protocols.

▶ **Proposition 60.** *Let $\mathcal{P}$ denote a population protocol and Φ a potential group for $\mathcal{P}$ of size e which is rapidly decreasing in all reachable configurations with at least m agents. Then $\mathcal{P}$ reaches a terminal configuration C after $\mathcal{O}(e(\sqrt{\alpha}|Q| + \alpha)\, n^2)$ random interactions in expectation.*

Proof. Let $\sigma = C_0C_1...$ denote a fair run of $\mathcal{P}$, and pick the smallest l s.t. C_l is terminal. We define

$$X_i^c := |\{j : \ \Phi_i \text{ is active at } C_j \text{ and } \Phi_i(C_j) - \Phi_i(C_l) = c\}|$$

We observe that $l \leq \sum_{i,c} X_i^c$ holds and will now proceed to prove a bound on the expected value $\mathbb{E}(X_i^c)$, for all i, c, if σ is generated via random interactions.

Consider $\mathbb{P}(X_i^c \geq k + 1 \mid X_i^c \geq k)$, for $k \geq 1$. We note that σ is generated by a (homogeneous) Markov chain and the index τ of the k-th configuration counting towards X_i is a stopping time. By the strong Markov property, the above probability is equal to the probability that $\mathcal{P}$ reaches some configuration C counting towards X_i^c when started in the configuration C_τ. This is at most $1 - \gamma$, where γ is the probability that $\mathcal{P}$ executes a transition reducing Φ_i at C_τ, as an active Φ_i cannot increase at any later point.

First, we know that Φ_i is active at C_τ, so some transition reducing Φ_i is enabled at C_τ and $\gamma \geq 1/n(n-1)$. However, if c is large enough we can get a better bound due to the fact that Φ_i is rapidly decreasing at C_τ.

Let $\delta_> \subseteq \delta$ denote the transitions reducing Φ_i. Let $\xi := \text{tmin}_t(C_\tau)$ for some $t = (q, p \mapsto q', p') \in \delta_>$. By definition we have $C_\tau(q), C_j(p) \geq \xi$ and thus the probability of executing t at C_τ is at least $\xi(\xi - 1)/n(n-1)$ (note that $q = p$ is possible). As $\xi(\xi - 1) \geq \xi^2/2 - 1$, we find $n(n-1)\gamma \geq \text{speed}_{\delta_>}(C_\tau) - |\delta_>|$.

Of course, $|\delta_>| \leq |Q|^2$, and rapidly decreasing implies $\text{speed}_{\delta_>}(C_\tau) \geq c^2/\alpha$. In total we get $\gamma \geq \max\{1, c^2/\alpha - |Q|^2\}/n(n-1)$.

From $\mathbb{P}(X_i^c \geq k + 1 \mid X_i^c \geq k) \leq 1 - \gamma$ for all $k \geq 1$ we get $\mathbb{E}(X_i^c) \leq 1/\gamma$ (similar to the geometric distribution). We then sum over i and c:

$$\mathbb{E}(l) \leq \sum_{i=1}^{e} \sum_{c=0}^{\infty} \mathbb{E}(X_i^c) \leq en(n-1) \sum_{c=0}^{\infty} \frac{1}{\max\{1, c^2/\alpha - |Q|^2\}}$$

We now use $c^2/\alpha - |Q|^2 \geq c^2/2\alpha$ for $c \geq \sqrt{2\alpha}|Q|$.

$$\leq en(n-1)\Big(\sqrt{2\alpha}|Q| + \sum_{c=1}^{\infty} \frac{2\alpha}{c^2}\Big) \leq en(n-1)\Big(\sqrt{2\alpha}|Q| + \frac{\alpha\pi^2}{3}\Big)$$

◀

I Miscellaneous

I.1 Refinement

To show correctness of the conversions, we use a notion of refinement; i.e. we show that the behaviour of our constructed population computer $\mathcal{P}'$ matches the behaviour of the original computer $\mathcal{P}$. Here, we define this notion formally and show that it implies that the computers are equivalent, in that they decide the same predicate.

▶ **Definition 61.** *Let $\mathcal{P} = (Q, \delta, I, O, H)$ and $\mathcal{P}' = (Q', \delta', I', O', H')$ denote population computers. We write $\mathcal{I}, \mathcal{I}'$ for the initial configurations for $\mathcal{P}$ and $\mathcal{P}'$, respectively.*

We say that $\mathcal{P}'$ refines *$\mathcal{P}$ if there is a $\pi : \mathbb{N}^{Q'} \to \mathbb{N}^{Q}$ with*
1. *$\pi(C) \to \pi(C')$ for all reachable $C, C' \in \mathbb{N}^{Q'}$ with $C \to C'$,*
2. *$I = I'$ and for $C \in \mathcal{I}'$ we have $\pi(C) \in \mathcal{I}$ and $\pi(C)(q) = C(q)$ for $q \in I$, and*
3. *$\pi(C)$ is terminal and $O(\text{supp}(\pi(C))) = O'(\text{supp}(C))$ for all terminal configurations $C \in \mathbb{N}^{Q'}$.*

To show that $\mathcal{P}'$ decides the same predicate as $\mathcal{P}$, we need the additional assumption that $\mathcal{P}'$ is terminating. (Recall that a population computer is terminating if every fair run is finite, while it is bounded if every run is finite.) Must of our constructions are bounded, and are thus terminating as well.

▶ **Lemma 62.** *Let $\mathcal{P}, \mathcal{P}'$ denote population computers, where $\mathcal{P}$ decides φ and $\mathcal{P}'$ is terminating. If $\mathcal{P}'$ refines $\mathcal{P}$, then $\mathcal{P}'$ decides φ as well.*

Proof. Let $C_0 \in \mathbb{N}^{Q'}$ denote an arbitrary initial configuration of $\mathcal{P}'$. We decompose $C_0 =: C_I + C_H$ into an input configuration $C_I \in \mathbb{N}^{I'}$ and a helper configuration $C_H \in \mathbb{N}^{\text{supp}(H')}$, $C_H \geq H$. As $\mathcal{P}'$ is terminating, there is a terminal configurations C with $C_0 \to C$. It now suffices to show $O'(\text{supp}(C)) = \varphi(C_I)$. For this, we use properties 1-4 of the definition of refinement.

Property 2 implies that $\pi(C_0)$ is an initial configuration of $\mathcal{P}$. Further, $\pi(C_0) = C_I + D_H$, where C_I is the same as above and $D_H \in \mathbb{N}^H$, $D_H \geq H$ is a helper configuration of $\mathcal{P}$. So we now only need to show that $\mathcal{P}$ outputs $O'(\text{supp}(C))$ on C_I, i.e. that there is a terminal configuration $D \in \mathbb{N}^Q$ reachable from $\pi(C_0)$ with $O(\text{supp}(D)) = O'(\text{supp}(C))$.

We set $D := \pi(C)$. By property 1, $\pi(C_0) \to \pi(C)$, and by property 3 we get that $\pi(C)$ is terminal. Finally, property 3 also implies $O(\text{supp}(\pi(C))) = O'(\text{supp}(C))$. ◀

I.2 Properties of minima

▶ **Lemma 63.** *Let $x, y_1, ..., y_k \in \mathbb{R}_{\geq 0}$. Then $\min\{x, \sum_i y_i\} \leq \sum_i \min\{x, y_i\}$.*

Proof. If $x \geq \sum_i y_i$ then $x \geq y_i$ for all i, so $\min\{x, \sum_i y_i\} = \sum_i y_i = \sum_i \min\{x, y_i\}$. Otherwise, we assume $x \leq \sum_i y_i$ and get $\min\{x, \sum_i y_i\} = x$. If there is a j with $y_j > x$, then $x = \min\{x, y_j\} \leq \sum_i \min\{x, y_i\}$, else we have $\min\{x, y_i\} = y_i$ for all i and thus $\sum_i \min\{x, y_i\} = \sum_i y_i$, which is at least x by assumption. ◀

▶ **Corollary 64.** *Let $x_1, ..., x_k, y_1, ..., y_k \in \mathbb{R}_{\geq 0}$. Then $\min\{\sum_i x_i, \sum_i y_i\} \leq \sum_{i,j} \min\{x_i, y_j\}$.*

▶ **Lemma 65.** *Let $x_1, x_2, y_1, y_2 \in \mathbb{R}_{\geq 0}$. Then $\min\{x_1 + x_2, y_1 + y_2\} \leq \min\{x_1, y_1\} + x_2 + y_2$.*

Proof. Both $\min\{x_1 + x_2, y_1 + y_2\} \leq x_1 + x_2 + y_2$ and $\min\{x_1 + x_2, y_1 + y_2\} \leq y_1 + x_2 + y_2$ are trivial, and the statement follows. ◀

I.3 Sum of squares inequality

The following inequality is well-known. Intuitively, it states that a sum of squares is minimised when all summands are equal (assuming that their sum is held constant).

▶ **Lemma 66.** *Let* $x_1, ..., x_n \in \mathbb{R}$. *Then*

$$\Big(\sum_i x_i\Big)^2 \leq n \sum_i x_i^2$$

Proof. We apply the Cauchy-Bunyakovsky-Schwarz inequality, yielding

$$\Big(\sum_{i=1}^{n} 1 \cdot x_i\Big)^2 \leq \Big(\sum_{i=1}^{n} 1^2\Big) \cdot \Big(\sum_{i=1}^{n} x_i^2\Big) = n \sum_{i=1}^{n} x_i^2$$

◀

References

1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in population protocols. In *SODA*, pages 2560–2579. SIAM, 2017.

2 Dan Alistarh and Rati Gelashvili. Recent algorithmic advances in population protocols. *SIGACT News*, 49(3):63–73, 2018.

3 Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *PODC*, pages 47–56. ACM, 2015.

4 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC*, pages 290–299. ACM, 2004.

5 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006.

6 Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Comput.*, 21(3):183–199, 2008.

7 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Comput.*, 20(4):279–304, 2007.

8 Michael Blondin, Javier Esparza, Blaise Genest, Martin Helfrich, and Stefan Jaax. Succinct population protocols for Presburger arithmetic. In *STACS*, volume 154 of *LIPIcs*, pages 40:1–40:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

9 Michael Blondin, Javier Esparza, and Stefan Jaax. Large flocks of small birds: On the minimal size of population protocols. In *STACS*, volume 96 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

10 Robert Brijder, David Doty, and David Soloveichik. Democratic, existential, and consensus-based output conventions in stable computation by chemical reaction networks. *Natural Computing*, 17(1):97–108, 2018.

11 David Doty, Mahsa Eftekhari, Leszek Gasieniec, Eric E. Severson, Grzegorz Stachowiak, and Przemyslaw Uznanski. Brief announcement: A time and space optimal stable population protocol solving exact majority. In *PODC*, pages 77–80. ACM, 2021.

12 Robert Elsässer and Tomasz Radzik. Recent results in population protocols for exact majority and leader election. *Bull. EATCS*, 126, 2018.

13 Christoph Haase. A survival guide to Presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018.